



US006111883A

United States Patent [19]

Terada et al.

[11] Patent Number: **6,111,883**[45] Date of Patent: **Aug. 29, 2000**[54] REPEATER AND NETWORK SYSTEM
UTILIZING THE SAME[75] Inventors: Masato Terada, Sagamihara; Makoto
Kayashima, Yamato; Takahiko
Kawashima, Yokohama; Tetsuya
Fujiyama, Yokohama; Minoru
Kotzum, Yokohama; Kazuo
Nishimura, Hadano; Kazunari
Hirayama; Takaaki Ogino, both of
Yokohama, all of Japan

[73] Assignee: Hitachi, Ltd., Tokyo, Japan

[21] Appl. No.: 08/884,133

[22] Filed: Jun. 27, 1997

[30] Foreign Application Priority Data

Jul. 12, 1996 [JP] Japan 8-182975
Oct. 18, 1996 [JP] Japan 8-275809[51] Int. Cl.⁷ G06F 13/38[52] U.S. Cl. 370/401; 370/389; 713/201;
709/229; 709/249[58] Field of Search 370/389, 390,
370/392, 400, 401, 402, 410; 395/186,
187.01, 200.33, 200.54, 200.57, 200.59,
200.79

[56] References Cited

U.S. PATENT DOCUMENTS

5,623,601 4/1997 Vu 395/187.01
5,689,566 11/1997 Nguyen 380/25
5,699,513 12/1997 Feigen et al. 395/187.01
5,793,763 8/1998 Mayes et al. 370/3895,802,320 9/1998 Bachr et al. 395/200.79
5,826,014 10/1998 Coley et al. 395/187.01
5,835,726 11/1998 Shwed et al. 395/200.57
5,864,683 1/1999 Boebert et al. 395/200.79
5,960,177 9/1999 Tanno 395/200.59

OTHER PUBLICATIONS

"Check Point Fire Wall-1™ White Paper", Version 3.0, Jan.
1997, P/N 400-3000.Primary Examiner—Chi H. Pham
Assistant Examiner—Frank Duong
Attorney, Agent, or Firm—Antonelli, Terry, Stout & Kraus,
LLP

[57] ABSTRACT

In view of providing a network system enabling communication having passed fire walls (repeaters) and assuring high security and operation flexibility through access control based on users and applications, a user-held table indicating correspondence between repeaters and passwords, a repeater-held table indicating correspondence between users and passwords and a table indicating access regions are defined respectively for users, departments of users and official positions of users and a route control information storing table indicating correspondence between networks and next transmitting destination is also provided to execute the access control for each user. Moreover, the repeater is provided with the repeating route control table so that a repeater located in the course of route to the transmitting destination computer and allowing communication from the transmitting side computer is selected from the data repeating control table and the process for requesting the repeating operation of communication with the destination is executed to the selected repeater.

32 Claims, 21 Drawing Sheets

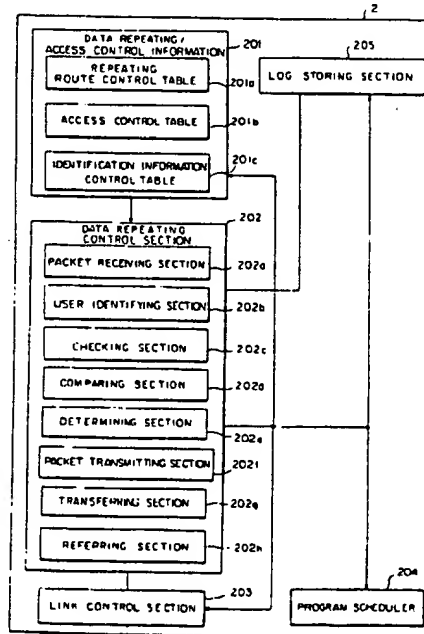
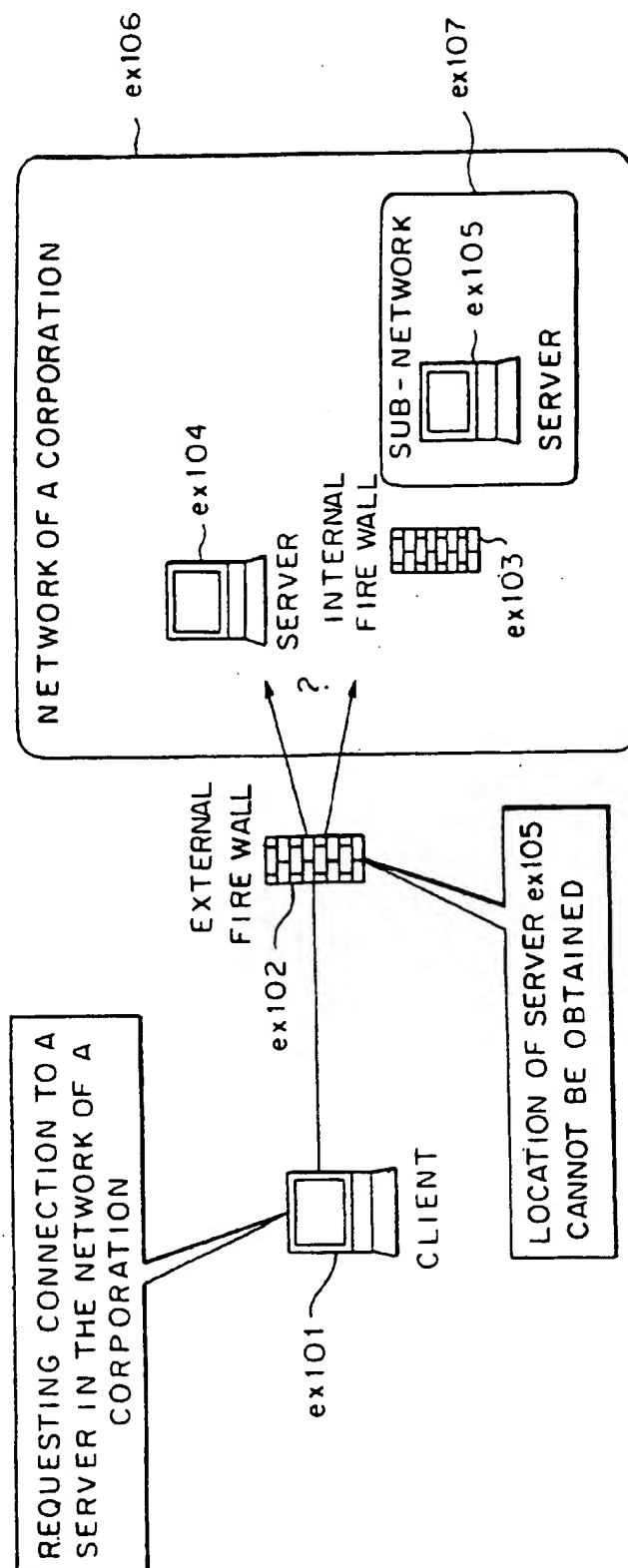


FIG. 1
PRIOR ART



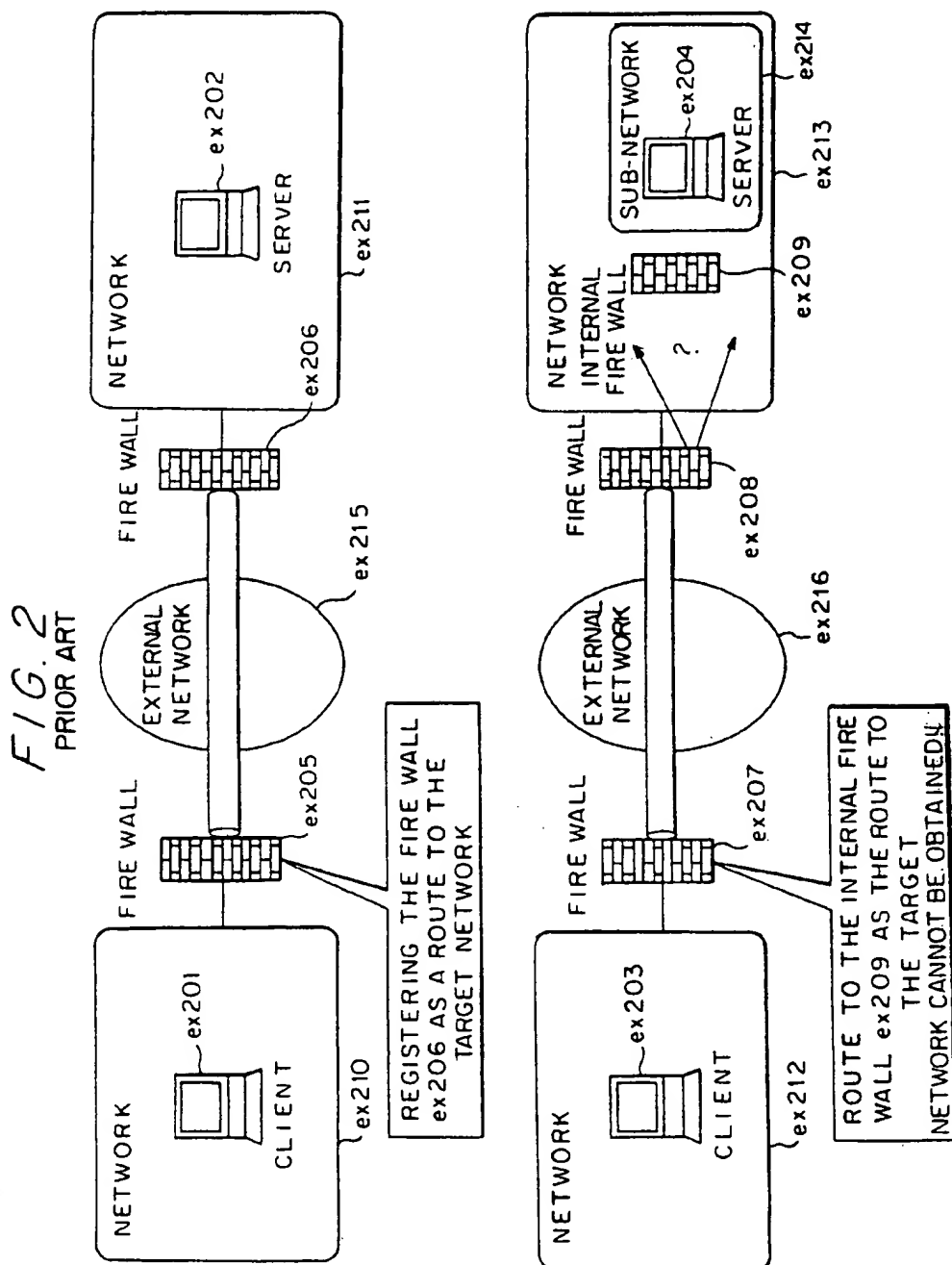


FIG. 3

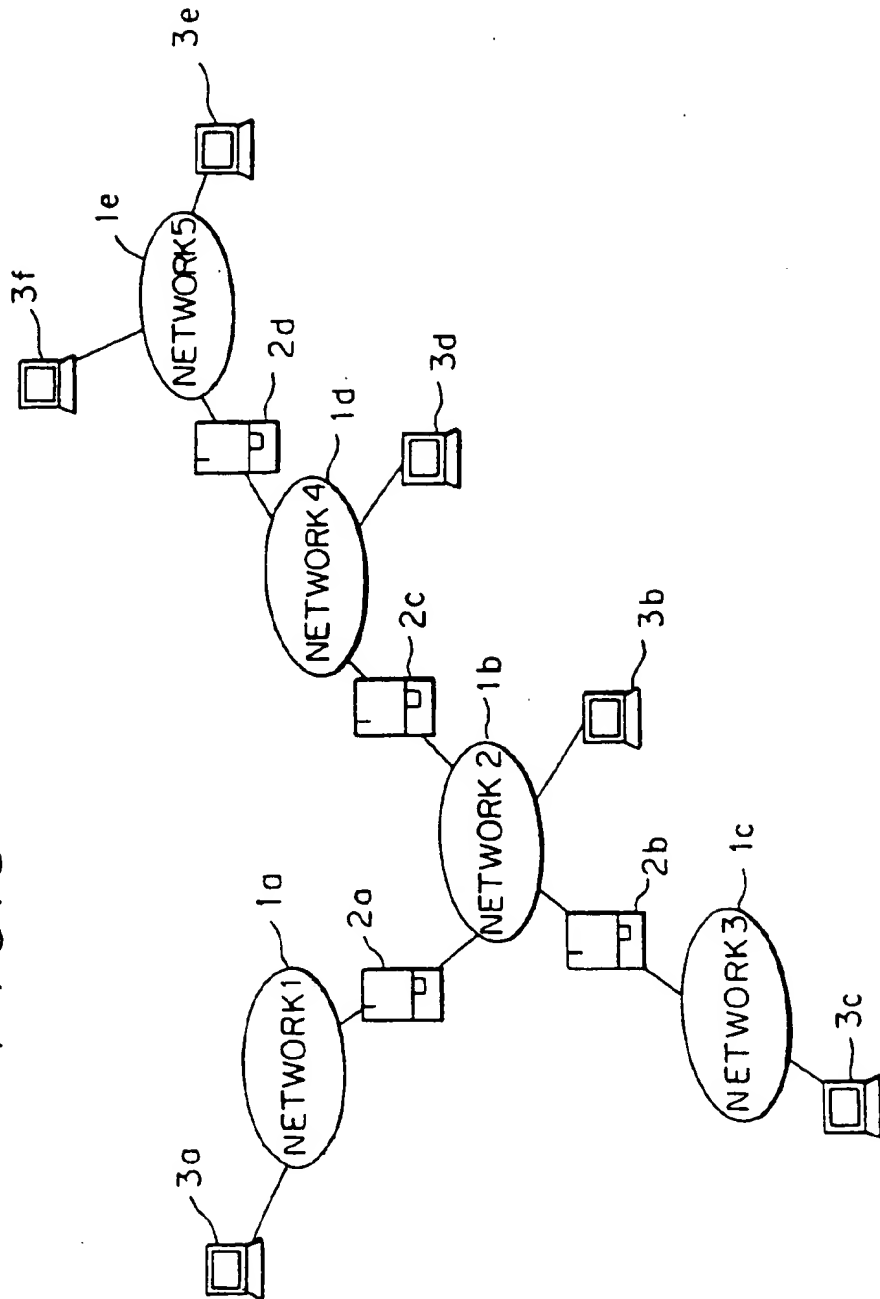


FIG. 4

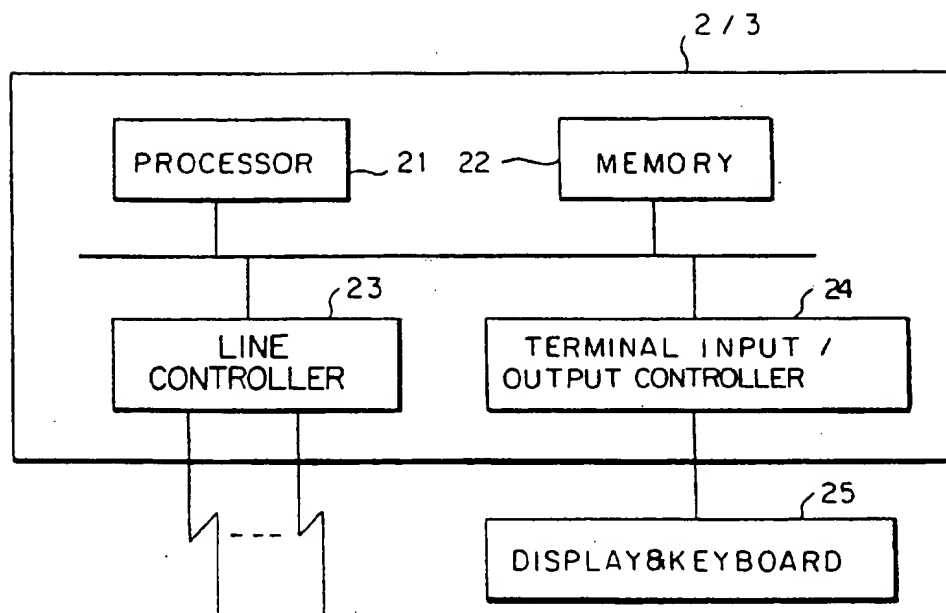
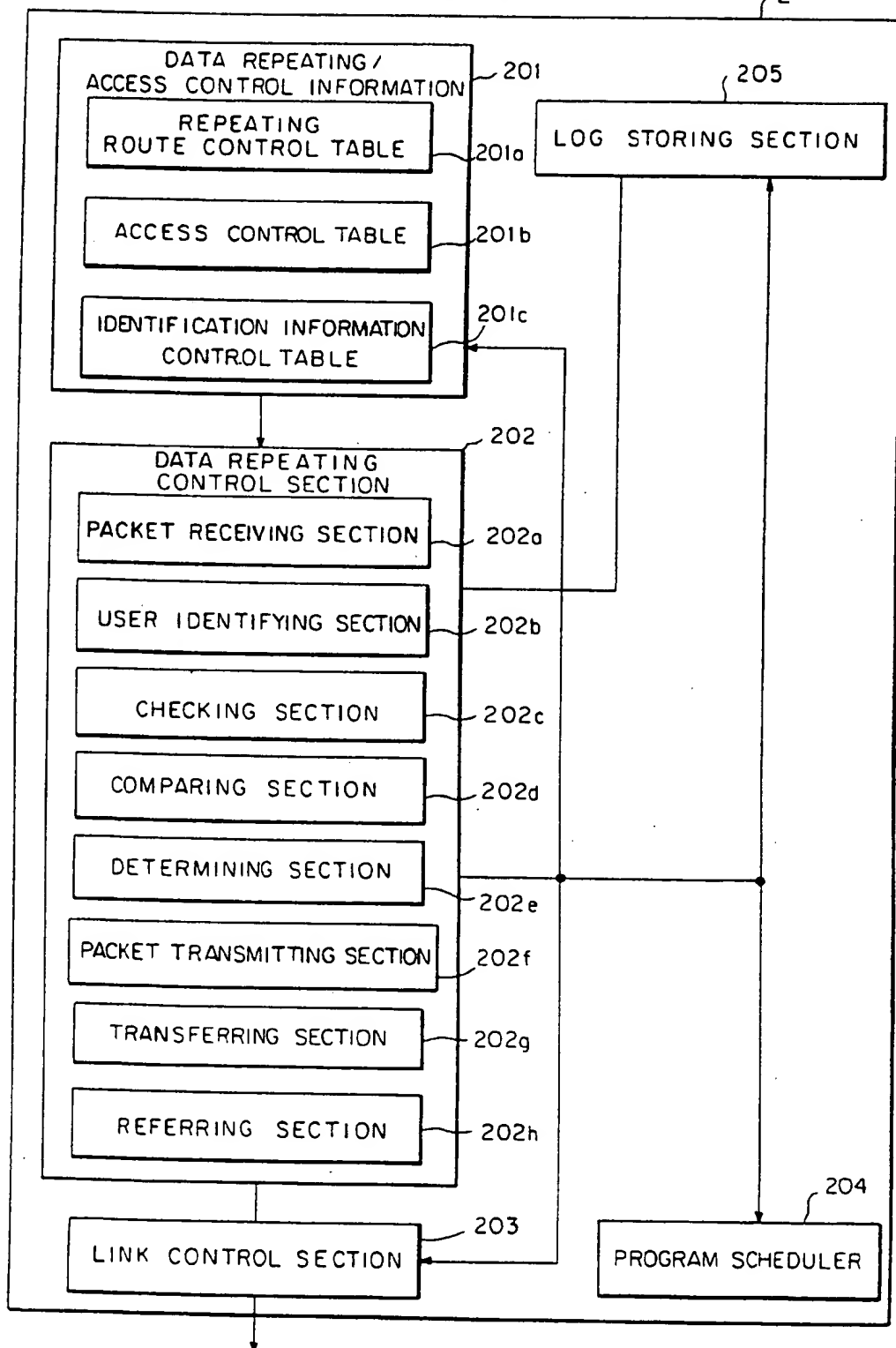


FIG. 5



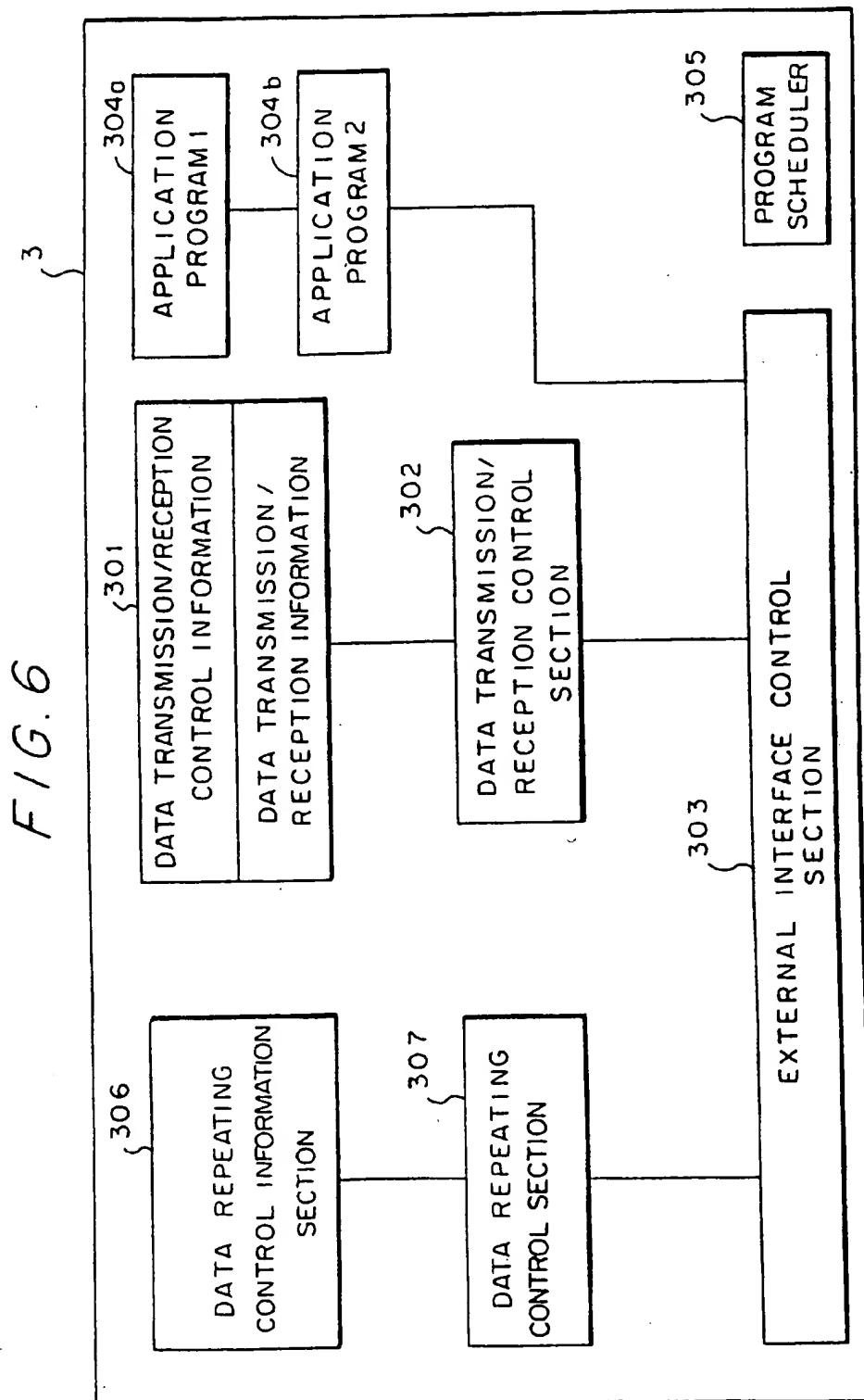


FIG. 7(A)

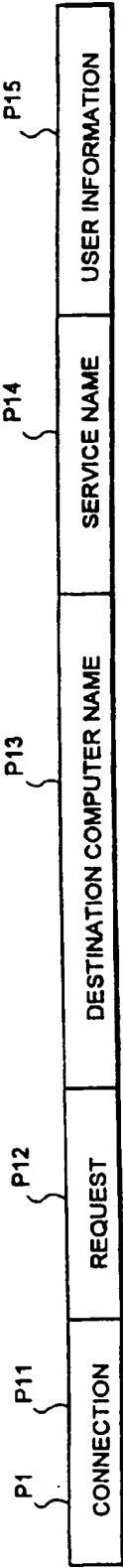


FIG. 7(B)

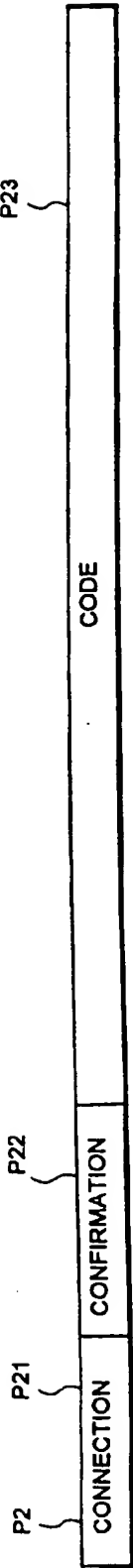


FIG. 7(C)



FIG. 8

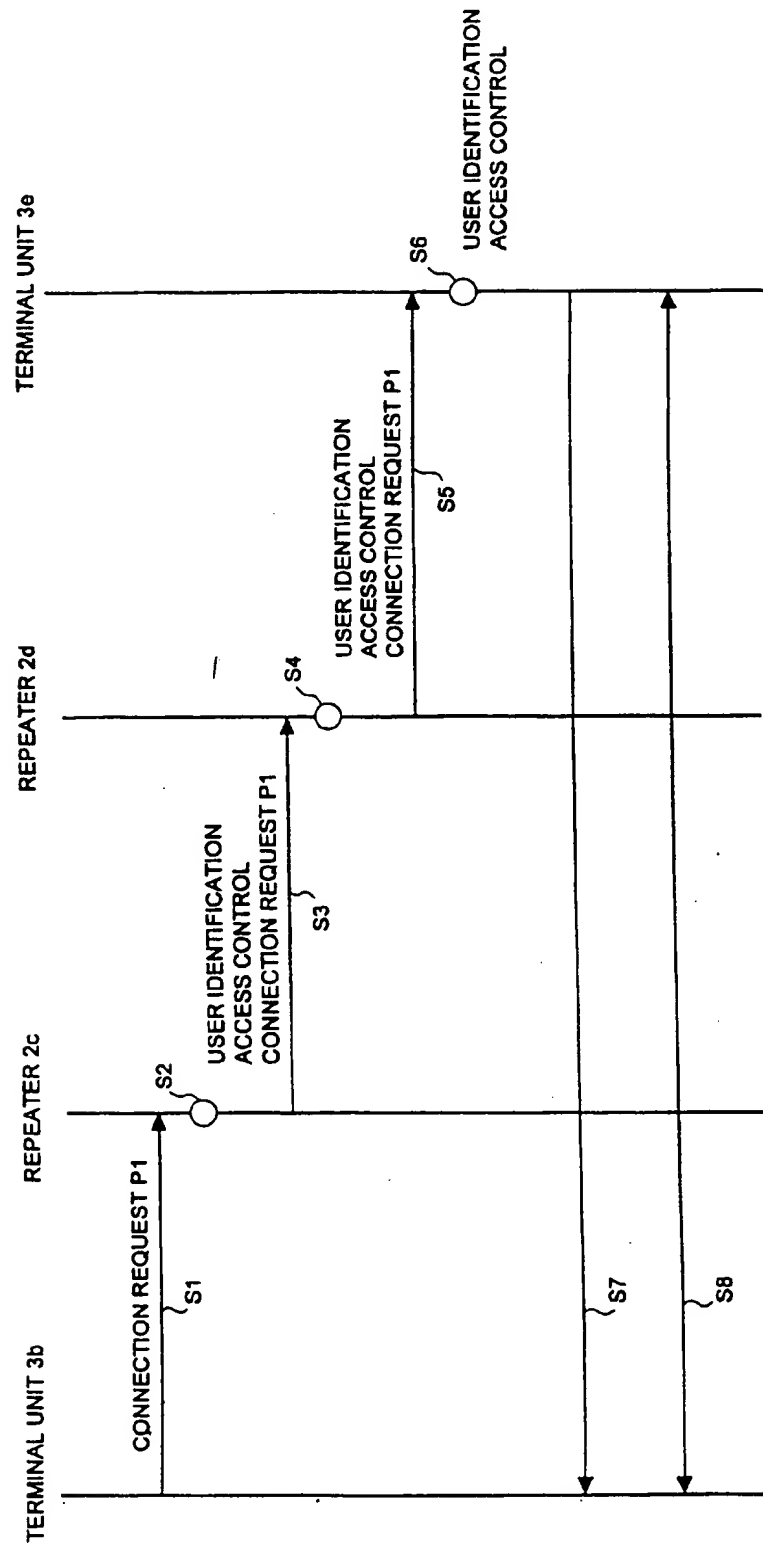
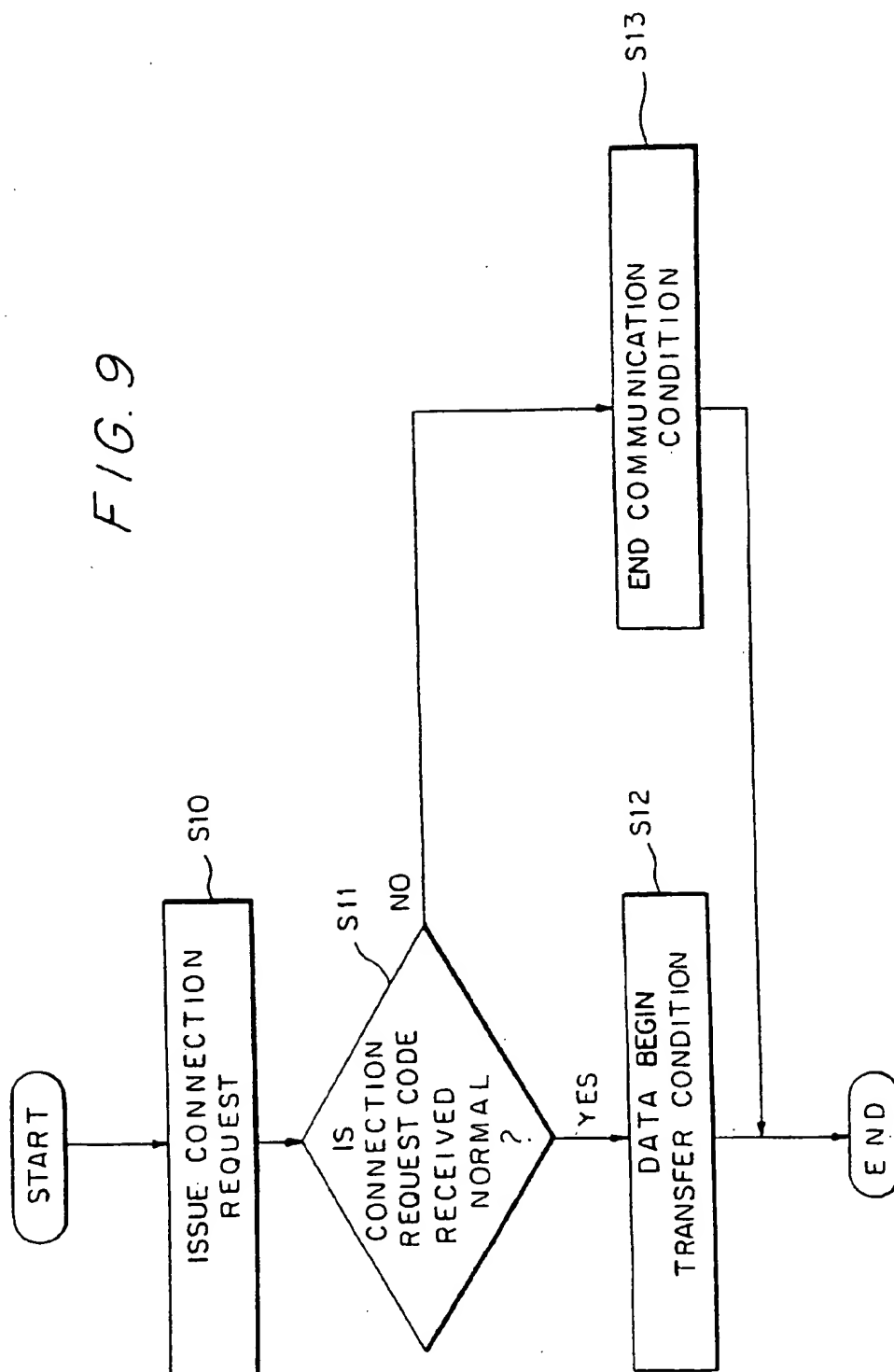


FIG. 9



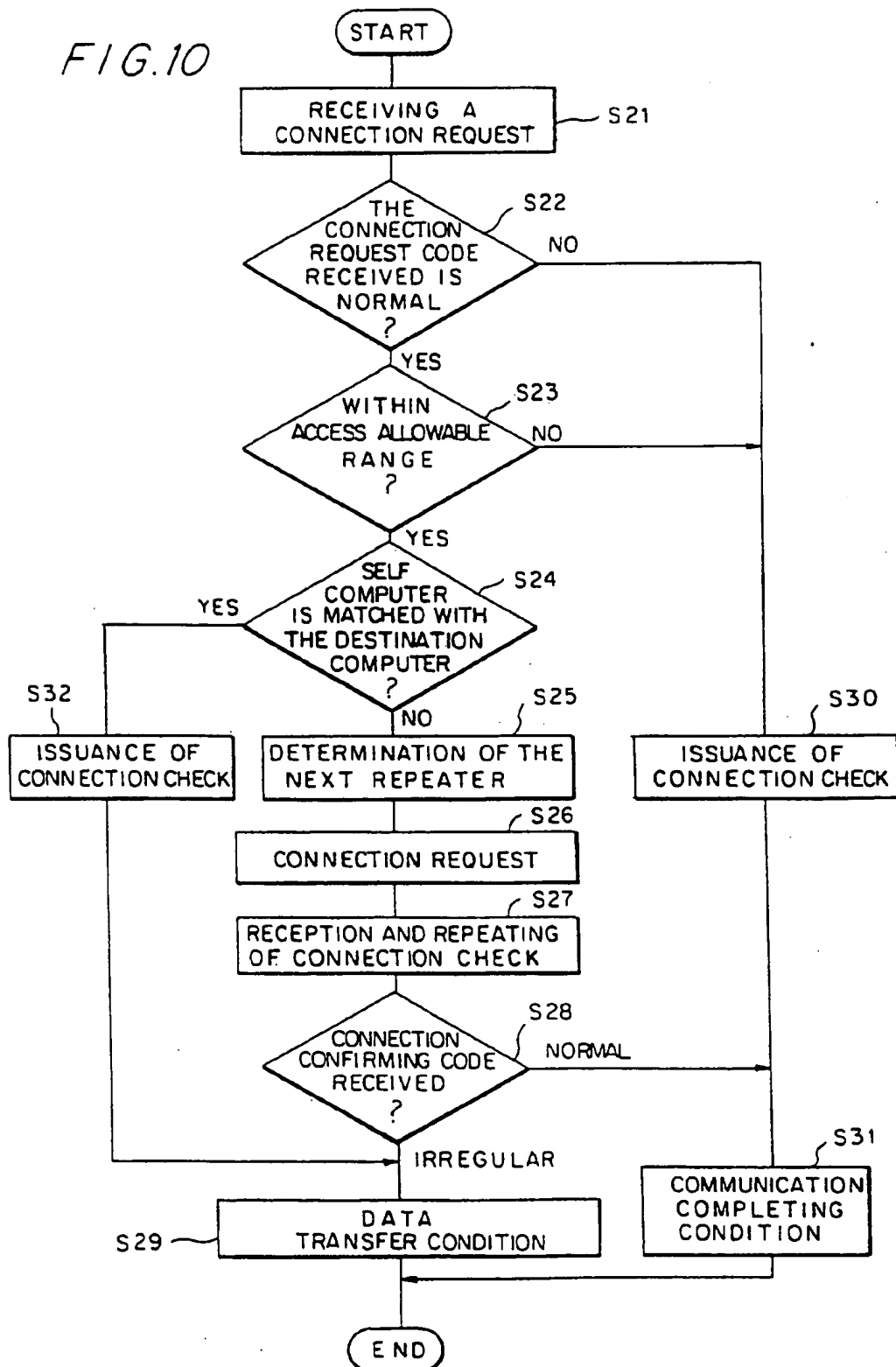


FIG. 11

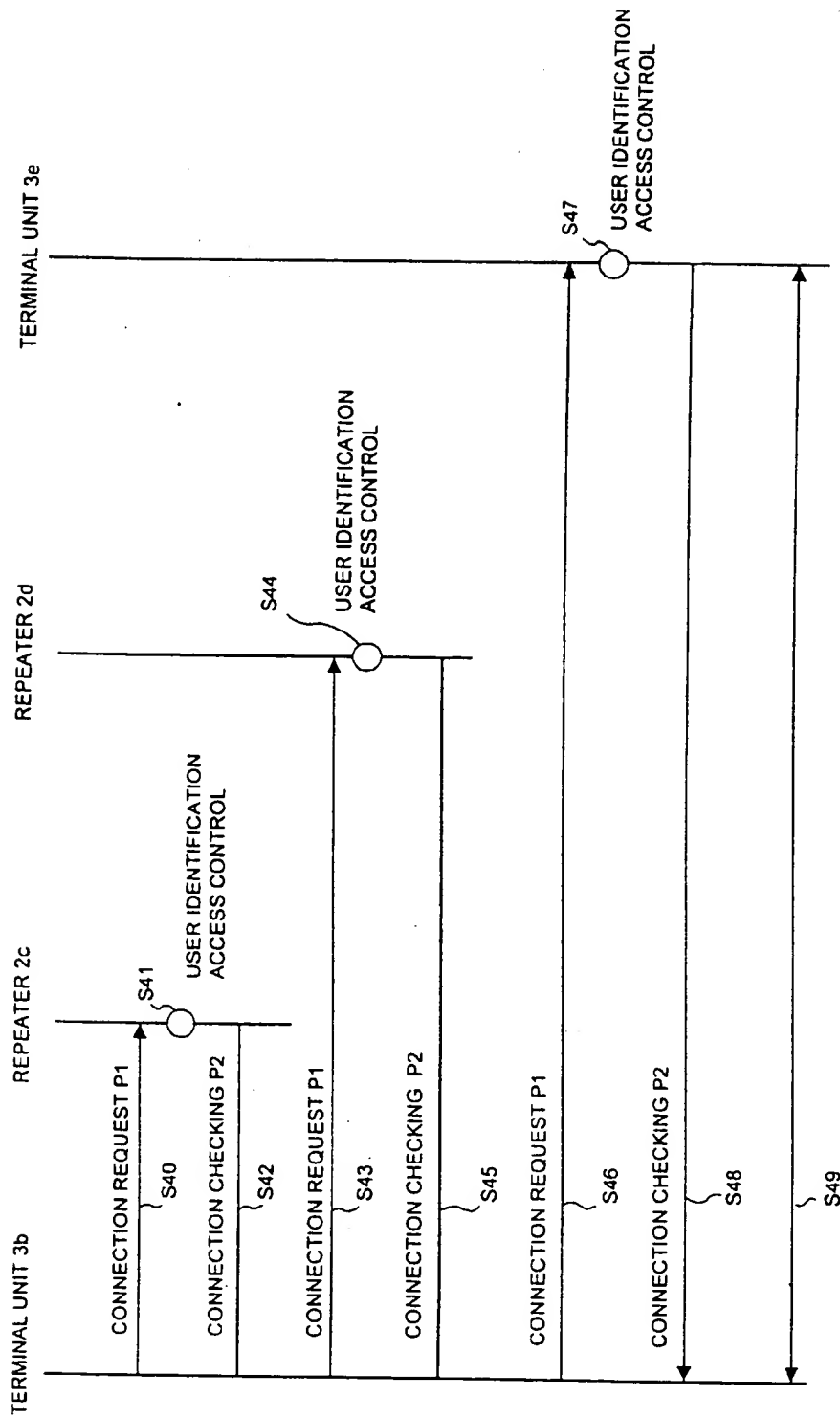


FIG. 12

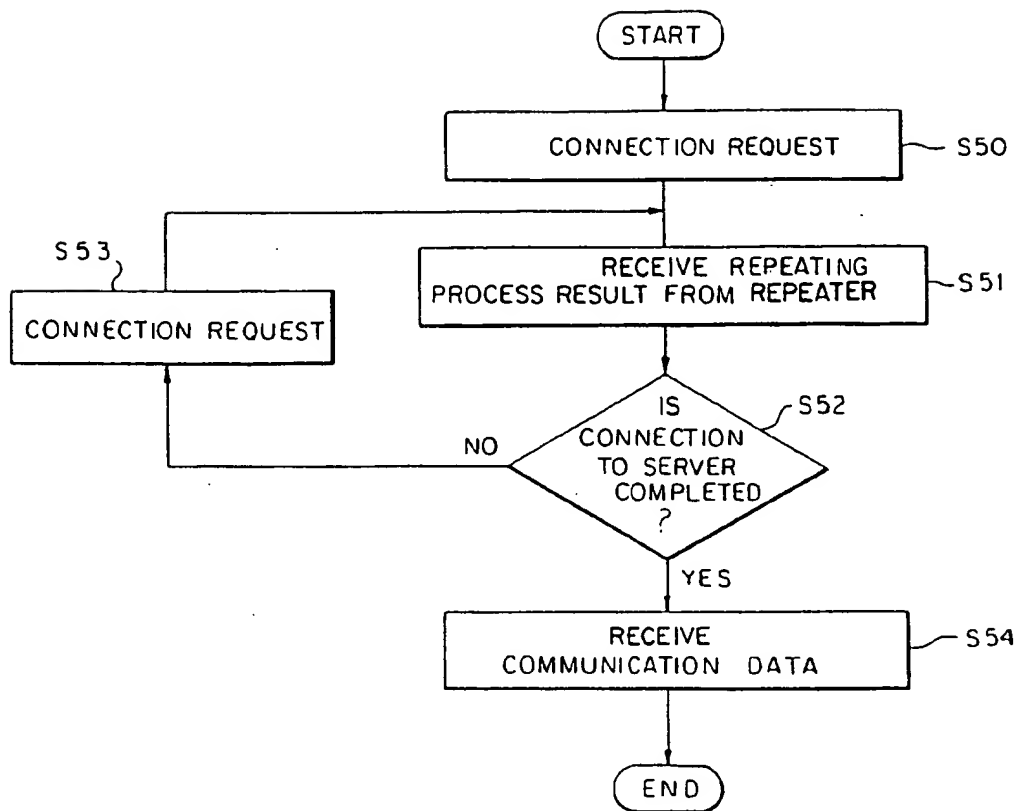


FIG. 13

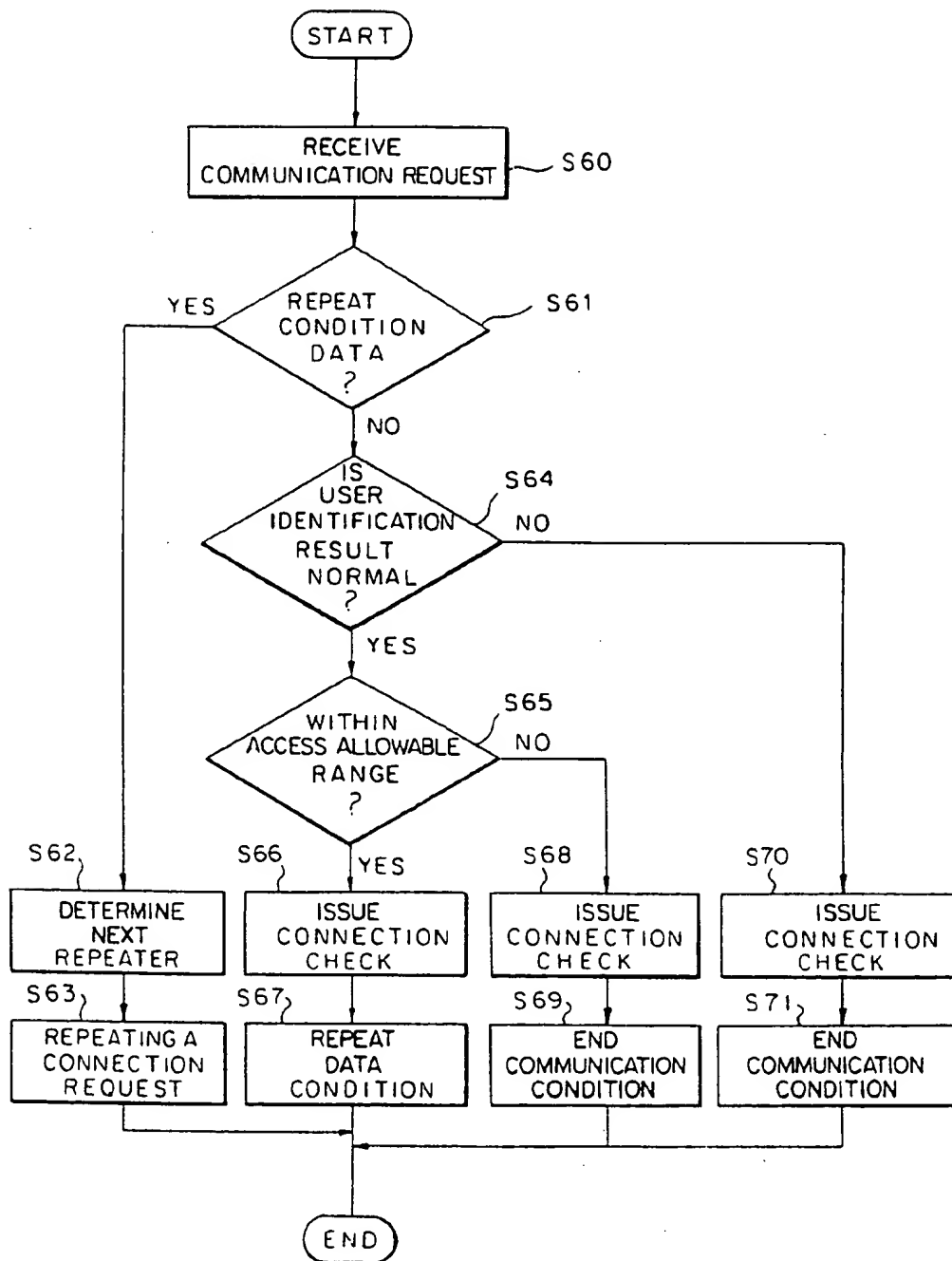


FIG. 14

REPEATER NAME	IDENTIFICATION INFORMATION
REPEATER a	TEST
REPEATER b	-
REPEATER c	-
REPEATER d	-

FIG. 15

USER NAME	IDENTIFICATION INFORMATION
USER 1	test
USER 2	abcdx
USER 3	poids
USER 4	odksic

FIG. 16

USER NAME	DEPARTMENT	OFFICIAL POSITION	TRANSMITTING SIDE	DESTINATION	SERVICE
427a USER 1	GENERAL AFFAIRS	GENERAL MANAGER	NETWORK 1 NETWORK 2	NETWORK 1 NETWORK 2	FILE TRANSFER
427b USER 2	-	SECTION CHIEF	NETWORK 3 NETWORK 5	NETWORK 3 NETWORK 5	-
427c USER 3	PLANNING	GENERAL MANAGER	-	-	VIRTUAL TERMINAL UNIT
427d USER 4	PLANNING	MEMBER	-	-	-

FIG. 17

431 DEPARTMENT	432 DESTINATION	433 TRANSMITTING SIDE	434 SERVICE
PLANNING	NETWORK 2 NETWORK 3 NETWORK 4 NETWORK 5	NETWORK 2 NETWORK 4	VIRTUAL TERMINAL UNIT
DEVELOPMENT	NETWORK 2 NETWORK 3 NETWORK 5	NETWORK 2 NETWORK 3	
DESIGN	NETWORK 2 NETWORK 3 NETWORK 5	NETWORK 2 NETWORK 3	VIRTUAL TERMINAL UNIT FILE TRANSFER
GENERAL AFFAIRS	NETWORK 1 NETWORK 2	NETWORK 1 NETWORK 2	DATABASE ACCESS

FIG. 18

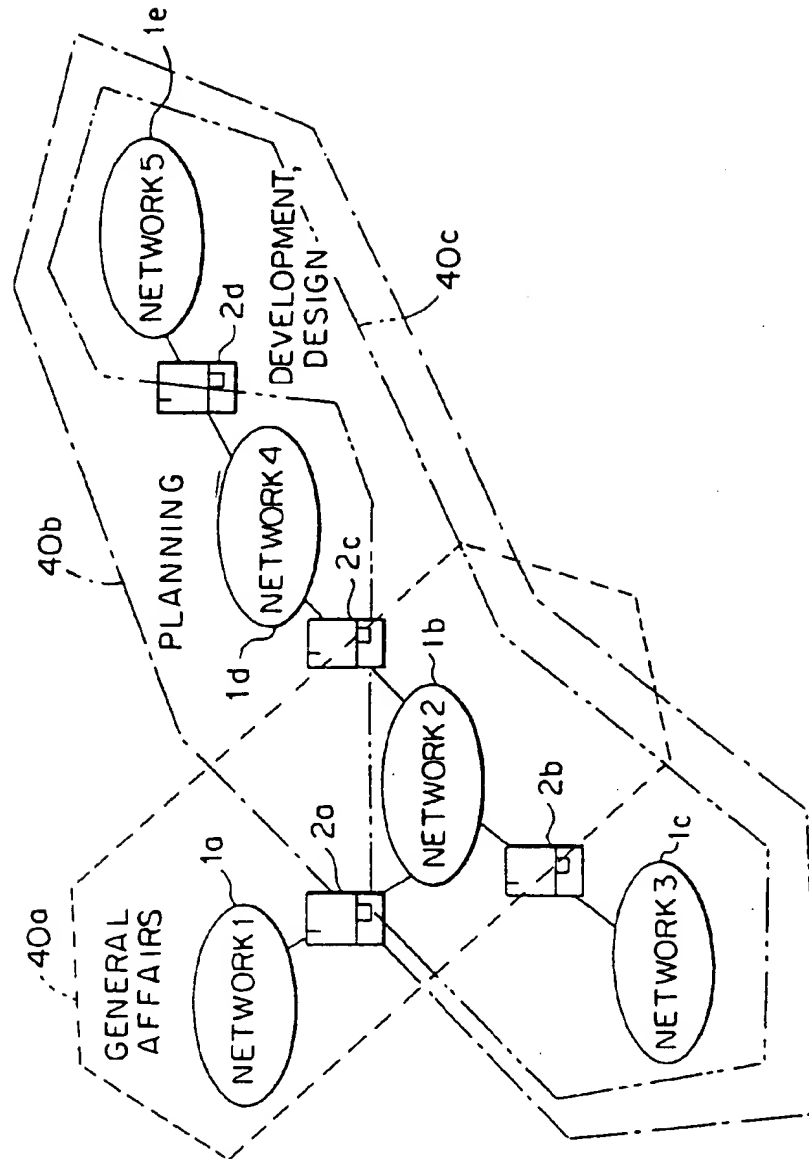


FIG. 19

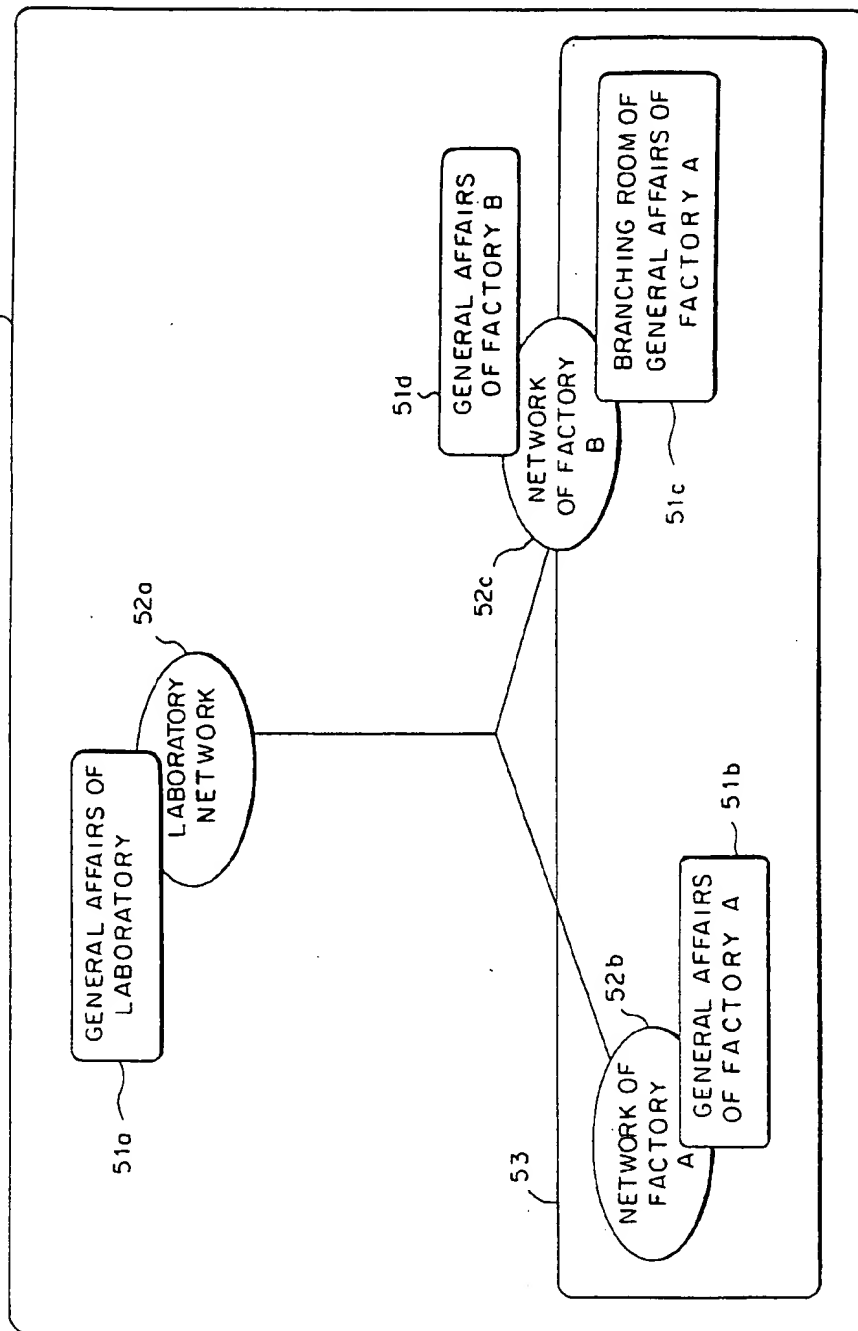


FIG. 20

	441 OFFICIAL POSITION	442 CLASS OF DESTINATION	443 REMOTE DESTINATION	444 SERVICE
445a	GENERAL MANAGER	LOCAL REMOTE	*	*
445b	SECTION CHIEF	LOCAL REMOTE	*	VIRTUAL TERMINAL UNIT
445c	CHIEF	LOCAL	-	-
445d	MEMBER	LOCAL	-	-

FIG. 21

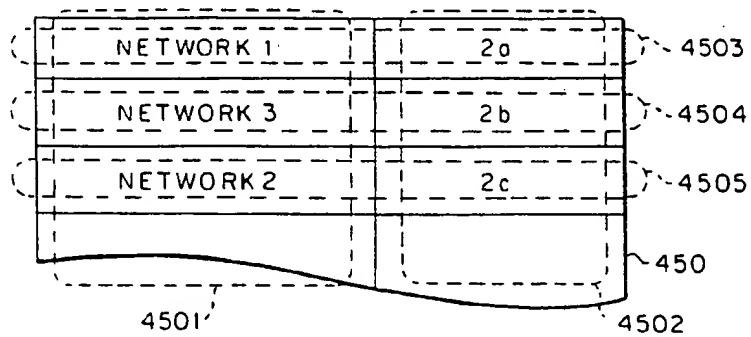


FIG. 22

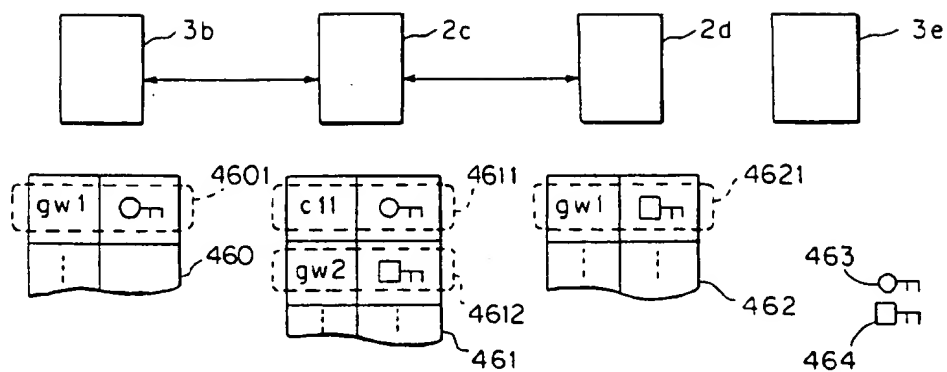


FIG. 23

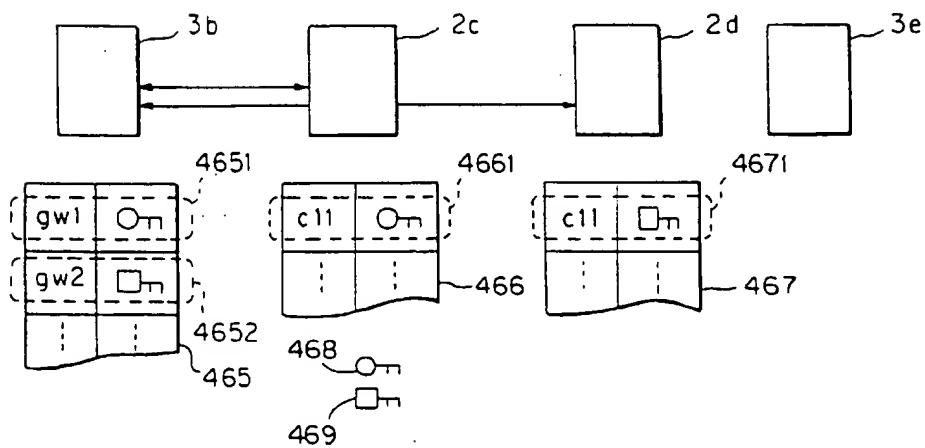


FIG. 24a

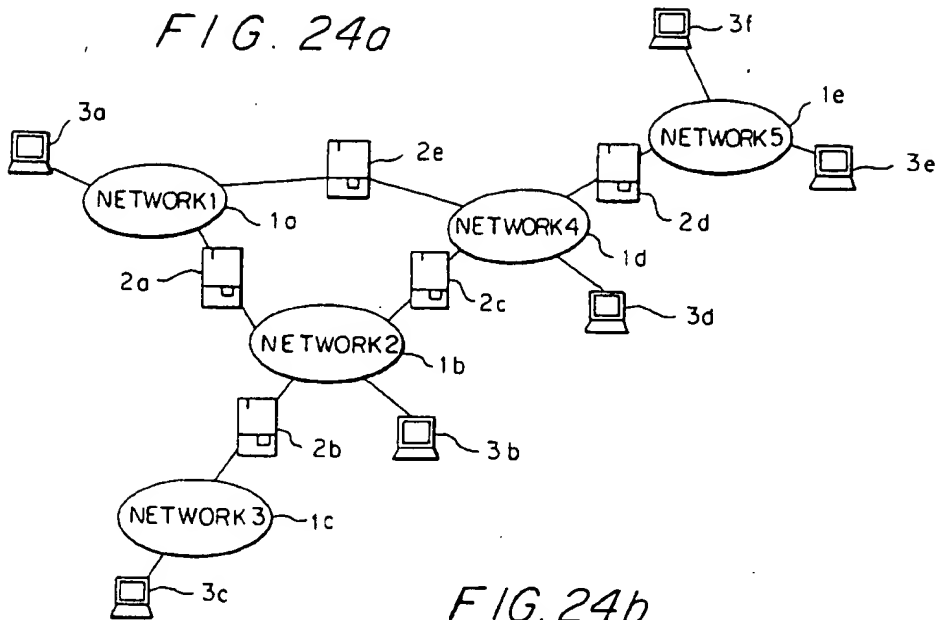


FIG. 24b

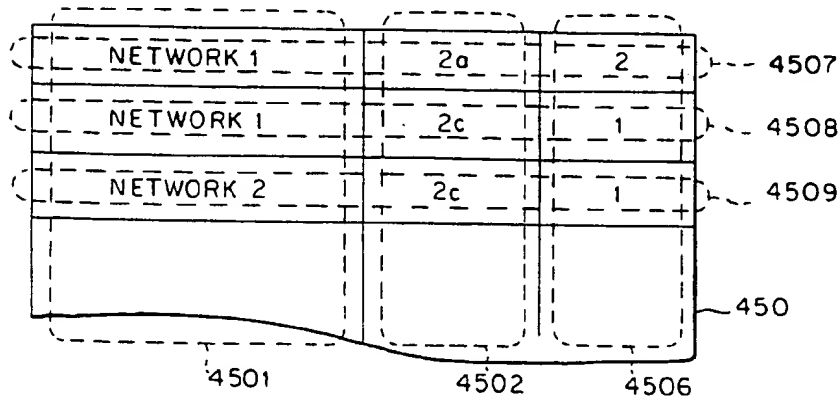


FIG. 25

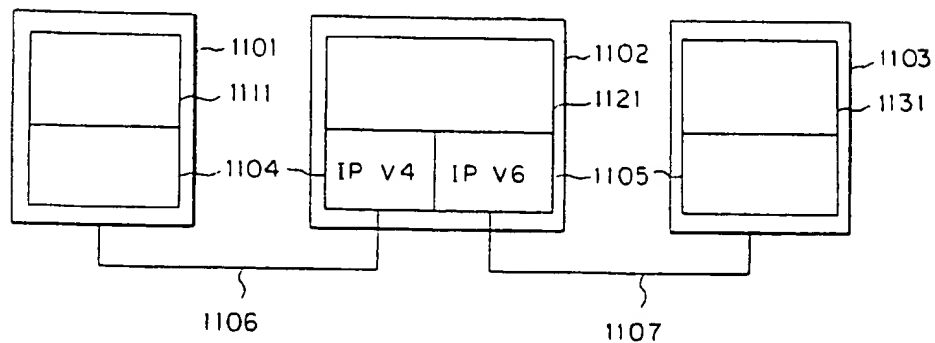


FIG. 26

471 {	472 {	473 {	474 {	475 {	476 {	477 {	470 {
USER NAME	TRANSMITTING SIDE	DESTINATION	SERVICE NAME	CONDITION	POSSIBILITY OF ACCESS	TIME	
USER 1	TERMINAL b	TERMINAL e	VIRTUAL TERMINAL UNIT	START	IMPOSSIBLE	1998-2-20 15:30	
USER 1	TERMINAL b	TERMINAL a	FILE TRANSFER	START	POSSIBLE	1998-2-21 18:00	
USER 1	TERMINAL b	TERMINAL a	FILE TRANSFER	END		1998-2-21 17:00	

REPEATER AND NETWORK SYSTEM UTILIZING THE SAME

BACKGROUND OF THE INVENTION

The present invention relates to security of a computer connected to a network system and particularly to a method of constituting a network system which executes access control and relays communications of applications through mutual cooperation of fire walls.

As a method of preventing invasion into a computer through a network, a repeater (fire wall) has been proposed to give restriction to the access from outside.

A typical fire wall has a function, as is described "Computer Security Resource Clearinghouse" of NIST (National Institute of Standards and Technology), to control the accesses depending on IP (Internet Protocol) addresses of the transmitting side and receiving side and kinds of services and to the store access record.

Moreover, as a repeater for repeating communication between a client and a server, there is provided socks V5 proposed by RFC1928 in the environment where fire walls exists. In the socks, mutual identification between the client and the repeating server and socks protocol for realizing connection instruction for the repeating server are defined and thereby communication between the client and the server having passed one fire wall can be realized.

Moreover, there is a gateway protocol such as RIP (Routing Information Protocol: RFC 1058), OSPF (Open Shortest Path First: RFC 1131), etc. as a mechanism to realize dynamic exchange of repeating route information in the IP layer.

With rapid development of Internet system, a person can get various kinds of information generated in the world on the real-time basis but, on the other hand, a person is in turn threatened to external invasion. As effective measures for such external invasion, it has been proposed to (1) give limitation on IP address for making access to each service and to (2) provide a gateway (fire wall in narrow sense) to store the access record. Use of such fire wall in narrow sense has enabled reduction of threat for an external invader by acquiring matching property of the operating environment of the gateway itself and localizing the range of control by an administrator.

However, in the case of executing the access control utilizing the technique of the related art, since the access control object is based on the information incorporated to a computer such as class of service and IP address, there is a problem that the access control based on users cannot be realized. For example, desired access control becomes impossible for the computer to which the IP address is assigned dynamically and class of service is limited to particular users.

Moreover, in private network utilizing the Internet, a fire wall plays a very important role for security and an internal fire wall is increasingly installed in the private network in order to protect the sub-network. There are several problems to be solved for the communication in the environment where a plurality of fire walls exist. For example, when the communication having passed the internal fire wall for protecting the sub-network is to be attempted from a computer of an external network, the communication must be repeated between the external fire wall and the internal fire wall.

However, since the routing information for the internal fire wall provided for repeating is concealed to the external

network, such routing information must be obtained with a certain method. FIG. 1 shows an example of the problem explained above. When a client ex101 attempts to make communication with a server accommodated in the network ex106 of A corporation, an external fire wall ex102 repeats the communication. Since the external fire wall ex102 can obtain the routing information to the server ex104 for communication with the server ex104 in the network ex106 of A corporation, communication can be repeated. However, since the server ex105 is concealed by the internal fire wall ex103 for the communication with the server ex105 accommodated in the sub-network ex107, the external fire wall ex102 cannot obtain the routing information to the server ex105 and thereby this communication cannot be repeated.

Moreover, in the case of the communication between two networks connected through the external network, this communication cannot be realized between respective internal fire walls, unless the routing information for identifying the internal fire wall is set for the external fire wall.

FIG. 2 shows an example of the problem explained above. A client ex201 accommodated in the network ex210 is capable of making communication with a server ex202 in the network ex211 by registering the fire wall ex206 as the route to the server ex202 in the fire wall ex205. However, when a server ex204 is provided in the internal sub-network ex214 of the network ex213, since the route is concealed by the fire wall ex208, the internal fire wall ex209 cannot be registered in the fire wall ex207.

OBJECT AND SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a large scale network system which enables communications having passed the fire wall and repeaters (fire walls) used in the same network by solving the problems explained above and offering a means for exchanging the repeating route information among a plurality of repeaters (fire walls).

Moreover, it is also an object of the present invention to provide a network system which enhances security and assures higher operation flexibility and repeaters used therein through the access control based on the computer users and applications.

The objects explained above will be achieved using following means.

(1) Access control based on computer users and applications
Executing access control as an object of access control on the basis of computer users and applications

(2) Identification of computer users and applications
Identifying, for executing access control, that the communication is requested by a person who has issued the request.

(3) Data transfer in the repeaters having the access control function

Providing transparency of communication in the communication between computers having the access control functions

The data transfer by the repeaters can be realized by providing, in the repeater, a repeating route control table storing correspondence between the address of the transmitting side computer and the address of the repeater provided to transfer the data to such address and executing the processing to select, from the data repeating route control table, the repeater provided in the course of the route to the target computer in the receiving side to enable the communication from the computer of the transmitting side and the processing to connect the repeating program of the repeater identified by the processing explained above to request the repeating of communication with the receiving side to the repeater.

BRIEF DESCRIPTION OF THE DRAWINGS

While the present invention has been described in detail and pictorially in the accompanying drawings it is not limited to such details since many changes and modifications recognizable to those of ordinary skill in the art may be made to the invention without departing from the spirit and the scope thereof. Other objects and advantages of the present invention will be apparent from the following detailed description of the presently preferred embodiments thereof, which description should be considered in conjunction with the accompanying drawings in which:

FIG. 1 is a diagram (No. 1) for explaining problems of the related art;

FIG. 2 is a diagram (No. 2) for explaining problems of the related art;

FIG. 3 is a diagram showing a structure of the network system as a whole;

FIG. 4 is a hardware block diagram;

FIG. 5 is a diagram showing a software structure of a repeater;

FIG. 6 is a diagram showing a software structure of a terminal unit;

FIG. 7 is a diagram showing a packet format;

FIG. 8 is a diagram showing the communication sequence 1;

FIG. 9 is a diagram showing a terminal unit control flowchart 1;

FIG. 10 is a diagram showing a repeater control flowchart 1;

FIG. 11 is a diagram showing the communication sequence 2;

FIG. 12 is a diagram showing a terminal unit control flowchart 2;

FIG. 13 is a diagram showing a repeater control flowchart 2;

FIG. 14 is a diagram showing a format of user identification information table;

FIG. 15 is a diagram showing a format of apparatus identification information table;

FIG. 16 is a diagram showing a format of user access control table;

FIG. 17 is a diagram showing a format of section access control table;

FIG. 18 is a diagram showing an example of accessible region;

FIG. 19 is a diagram showing an example of a hierarchical network structure;

FIG. 20 is a diagram showing a format of official position access control table;

FIG. 21 is a diagram showing a format of repeating path information table;

FIG. 22 is a diagram showing a mutual identification method 1;

FIG. 23 is a diagram showing a mutual identification method 2;

FIG. 24 is a diagram for explaining dynamic path control;

FIG. 25 is a diagram for explaining a protocol conversion function; and

FIG. 26 is a diagram showing a format of table storing application logs.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Preferred embodiments of the present invention will be explained below.

The network system as an object in this embodiment has following characteristics.

(a) For distribution of data packet among communication apparatuses, distribution functions such as TCP (Transmission Control Protocol)/IP (Internet Protocol), OSI (Open Systems Interconnection), etc. are used.

(b) For data transfer, a repeater having access control function is provided.

Next, the structure of this network system will be explained with reference to FIG. 3 to FIG. 6.

FIG. 3 shows an example of the structure of this network system.

The network system of the present invention has structure that a plurality of networks 1 accommodating terminal units 3 are connected via repeaters (fire wall) 2. In this system, the repeaters 2a to 2d are capable of processing the TCP/IP, OSI protocol, etc., has distribution function of OSI data packet and is also provided with the access control function. In the explanation of this embodiment, the repeater is described as a fire wall. The terminal units 3a to 3f are computers installed in each user site. The networks 1a to 1e mean the networks such as the LAN (Local Area Network) and private line, etc.

FIG. 4 shows the structure of repeater 2 as an example of the hardware structure of the repeater 2 and the terminal unit 3 of a user site. The repeater 2 includes a processor 21 for controlling hardwares, a memory 22 for storing programs and transmitting/receiving messages, a line controller 23 for controlling input and output of signals to/from LAN and private line and a terminal input/output controller 24 for controlling a display and a keyboard connected to the apparatus. The repeater 2 is connected with a display and keyboard 25 as input/output devices.

FIG. 5 shows the software structure of the repeater 2 formed depending on the hardware structure shown in FIG. 4.

The software of the repeater 2 includes a storing section 201 for storing the repeating control information and access control information for transferring and filtering data packet, a data repeating control section 202 for offering the function to transfer the data packet to the target terminal unit depending on the repeating control information and the filtering function to discard of the data packet, a link control section 203 provided in a line control section 23 and a terminal input and output control section 24 to work as an external interface control section to control input and output of the LAN and private line and the terminal unit, a program scheduler 204 for scheduling and administrating program execution of the storing section 201, the data repeating control section 202 and the link control section 203 and a log storing section 205 for storing user application log.

The above functions of the software of the repeater 2 is realized by the processing performed by the processor 21.

In addition, the software executed by the processor 21 is stored in the memory 22, for example.

The program may also be retrieved from a storage medium such as floppy, ROM, etc or from a storage of a server connected to a network which is connected to the repeater, and stored in the memory 22.

In the repeating control information being stored in the storing section 201, a destination address information of the terminal unit (position of terminal unit, terminal unit name, etc.) and a next transmitting address information for sending data to the destination address are registered. Moreover, in the access control information, a user name, various attributes of user (department, official position, available services and accessible range, etc.) are registered.

FIG. 6 shows the software structure of a terminal unit 3 formed depending on the hardware structure shown in FIG. 4.

The software of the terminal unit 3 includes a storing section 301 for storing data transmitting and receiving control information as the route information for transmitting and receiving data packet and data transmission and reception information, a data transmission and reception control section 302 for controlling transmission and reception of data packet to and from the target terminal unit depending on this route information, an external interface control section 303 provided in a line control section 23 and a terminal input and output control section 24 to control the input and output of the LAN and private line and terminal unit, a plurality of application programs 304a to 304b operating on the terminal unit 3, a program scheduler 305 for scheduling and administering program execution of the storing section 301, data transmission and reception control section 302, external interface control section 303 and application programs 304, a data repeating control information section 306 to determine the transmitting destination of the data packet stored in the storing section 306 and a data repeating control section 307 for offering the function to transmit the data packet to the target repeater depending on the data repeating control information.

The above functions of the software of the terminal unit 3 is realized by the processing performed by the processor 21.

In addition, the software executed by the processor 21 is stored in the memory 22, for example.

The program may also be retrieved from a storage medium such as floppy, ROM, etc or from a storage of a server connected to a network which is connected to the terminal unit, and stored in the memory 22.

Next, the packet format and outline of the transmission procedures are explained with reference to FIG. 7 to FIG. 12.

FIG. 7 shows an example of the packet format used in this embodiment. FIG. 7(A) shows a format of the connection request packet P1 for requesting start of communication, while FIG. 7(B) shows a format of the connection confirming packet P2 and FIG. 7(C) shows a format of the data transfer packet P3.

Each packet is writing a class of packet in the first field, an operating method in the second field and data in the third and subsequent fields. In the case of the connection request packet P1 for requesting start of communication, "CONNECT" is set to the first field P11, "req" is set in the second field P12 indicating the operating method. In regard to the third field P13 and subsequent fields for transferring data, "transmitting destination terminal unit name" is set in the third field P13, "service name" in the fourth field P14 and "user information" to the fifth field P15. In the user information field P15, the user identification information and transmitting side terminal unit name are stored.

In the connection confirming packet P2 indicating the response for start of communication, "CONNECT" is set to the first field P21, "conf" to the second field P22 and "code" to the third field P23. In the code third field P23, the codes indicating "allowing connection setup", "user identification error", "out of accessible range", etc. and information including names of repeater which has generated such codes and transmitting destination terminal unit are stored as the information indicating the condition of the communication starting operation.

In the data packet P3 used under the communicating condition, "DATA" is set to the first field P31, "null" to the second field P32 and "data" to the third field P33.

FIG. 8 shows the sequence of communication procedures by making access to the terminal unit 3e from the terminal unit 3b in the system shown in FIG. 3.

In this embodiment, prior to start of communication with the target terminal unit, the communication route is established using a packet for declaring start of communication. The connection request packet P1 is the packet for declaring start of communication. The terminal unit 3b transmits, prior to start of communication, the connection request packet P1 having designated the terminal unit 3e as the destination address of the target terminal unit in the third field P13 to the repeater 2c (S1).

In the repeater 2c, a user is identified depending on the user identification stored in the user information field P15 of the connection request packet P1 and thereafter it is judged whether a user is capable of using the repeater 2c or not (S2). When a user is judged to be capable of using the repeater, the connection request packet P1 received is transferred to the next repeater 2d in order to transmit the connection request packet P1 to the target terminal unit (S3). In the repeater 2d, when a user is also judged to use the repeater (S4) in the same manner as those for the repeater 2c, the connection request packet P1 is transmitted to the target terminal unit (S5).

In the terminal unit 3e, after a user is identified (S6), the connection confirming packet P2 having set the normal code "allowing connection setup" in the code field P23 is transmitted to the terminal unit 3b in the transmitting side as the response to the connection request packet P1 (S7). Thereby, the communication route is established between the terminal unit 3b and the terminal unit 3e and data communication may be started to transfer the data packet P3 (S8).

FIG. 9 shows a control flowchart for executing the communication start processing prior to start of communication by the terminal unit 3b with the target terminal unit. The connection request packet P1 designating the target terminal unit 3e in the destination terminal unit name field P13 is transmitted to the repeater 2c (S10). Upon reception of the connection confirming packet P2 as the communication route setup response packet, reference is made to the code field P23 of the connection confirming packet P2 (S11). When the code field P23 is normal, data transfer is started (S12) but if the code field P23 is irregular, communication is completed (S13).

FIG. 10 shows a control flowchart for executing communication start processing by the repeater 2c with terminal units.

When a packet receiving section 202a, included in the data repeating control section 202, receives the connection request packet P1 having designated the target terminal unit 3e as the destination (S21), user identifying section 202b, included in the data repeating control section 202, refers to the user information field P15 stored in the connection request packet P1 to identify a user (S22). When irregularity is not detected as the result of user identification, accessible range of user and matching between terminal units in the transmitting and receiving sides are checked by a checking section 202c, included in the data repeating control 202, that checks range and matching according to a user attribute table in the data repeating control information/access control information 201. The checking section 202c controls access to the terminal or service. The table stores correspondence between at least one attribute of at least one user and accessible range of networks. (S23). When the accessible range is satisfied, the destination terminal unit name field P13 of the connection request packet P1 is compared with the self terminal unit name as the repeating operation by a

comparing section 202d included in the data repeating control section 202 (S24). Since the repeater 2c is operating as a repeater and content of the destination terminal unit name field P13 does not match the self terminal unit name, a determining section 202e, included in the data repeating control section 202, determines the next repeating unit name with reference to a repeating route control table 201a in the data repeating control information/access control information 201 (S25). Next, a packet transmitting section 202f included in the data repeating control section 202 transmits the connection request packet P1 (S26). When the connection confirming packet P2 is received as the response of the connection request packet P1, the connection confirming packet P2 received is transferred to the terminal unit 3b which transmitted the connection request packet P1 by a transferring section 202g included in the data repeating control section 202 (S27). Moreover, reference is made to the code field P23 of the connection confirming packet P2 by a referring section 202h included in the data repeating control section 202 (S28). When the code field P23 is normal, data transfer is started (S29), but if the code field P23 is irregular, communication is completed (S31). If irregularity is detected as the result of user identification at step S22, the connection confirming packet P2 setting the error code "irregular user identification" in the code field P23 is transmitted to the terminal unit 3b which has transmitted the connection request packet P1 by the transmitting section 202f (S30) and the communication is completed (S31).

When output of accessible range is judged at step S23, the connection confirming packet P2 setting the error code "out of accessible range" in the code field P23 is transmitted to the terminal unit 3b which has transmitted the connection request packet P1 (S30) and communication is completed (S31).

This control flowchart includes the operations in the destination terminal unit. When the destination terminal unit name field P13 matches with the self terminal unit name at step S24, the self terminal unit is judged as the destination terminal unit in this control flowchart and the connection confirming packet P2 setting the normal code "allowing connection setup" in the code field P23 is transferred to the terminal unit 3b which has transmitted the connection request packet P1 (S32) to start the data transfer (S29).

FIG. 11 shows a modification example of the other embodiment of the communication procedure sequence for making access to the terminal unit 3e from the terminal unit 3b. In the example of sequence shown in FIG. 9, the connection request packet P1 is sequentially transferred by the repeaters, the repeaters must be in the reliable condition with each other. Meanwhile, the example of sequence in this embodiment indicates that the repeaters are not in the reliable condition with each other.

First, prior to start of communication with the target terminal unit, a communication route is established using the packet for declaring start of communication. The connection request packet P1 is the packet for declaring start of communication. A terminal unit 3b transmits, prior to start of communication, the connection request packet P1 designating the target terminal unit 3e as the destination to the repeater 2c (S40). In the repeater 2c, after user identification is performed depending on user identification stored in the user information field P15 of the connection request packet P1, a user is judged to be capable of using the repeater 2c or not (S41). When a user is judged to be capable of using the repeater, the connection confirming packet P2 is transmitted to the terminal unit 3b in the transmitting side (S42).

Upon reception of the connection confirming packet P2 from the repeater 2c, the terminal unit 3b transmits again the connection request packet P1 designating the target terminal unit 3e as the destination to the repeater 2c. The repeater 2c transfers in turn this connection request packet P1 to the repeater 2d (S43).

In the repeater 2d, when a user is judged to be capable of using the repeater 2d in the similar procedures as those for the repeater 2c (S44), the connection confirming packet P2 is transmitted to the terminal unit 3b of the transmitting side (S45).

The terminal unit 3b in the transmitting side transmits, upon reception of the connection confirming packet P2, the connection request packet P1 designating the target terminal unit as the destination to the repeater 2c. The repeaters 2c and 2d transfer this packet P1 to the target terminal unit 3e (S46).

The destination terminal unit 3e identifies a user depending on user identification stored in the user information field P15 of the connection request packet P1 (S47) and transmits the connection confirming packet P2 to the terminal unit 3b in the transmitting side as a response to the connection request packet P1 (S48). Thereby, the communication route can be set up between the terminal unit 3b in the transmitting side and the destination terminal unit 3e, data communication can be started and data packet P3 can be transmitted (S49). With execution of repeated communication route setup request, user identification for the terminal unit 3b in the transmitting side is performed for each repeater and services of this invention can also be offered even when reliable condition is not yet established among the repeaters.

FIG. 12 shows a control flowchart for executing the communication start processing prior to start of communication by the terminal unit 3b with the target terminal unit 3e. The connection request packet P1 designating the target terminal unit as the destination in the destination terminal unit name field P13 is transmitted to the repeater 2c (S50). Thereby, when the connection control packet P2 which is the communication route setup response packet is received in turn, whether connection to the target terminal unit 3e is completed or not is judged (S52) by referring to the code field P23 of the connection confirming packet P2. When the packet P2 is issued to confirm the connection from the repeater, the connection request packet P1 is transmitted again to the repeater 2c (S53) and operation returns to the step S51. When the packet P2 is issued to confirm the connection from the terminal unit 3e, data transfer is started (S54).

FIG. 13 shows a control flowchart for executing communication start process by the repeater 2c with a terminal unit depending on the sequence shown in FIG. 11. The repeater 2c starts, upon reception of the connecting request packet P1 (S60) designating the target terminal unit 3e as the destination, the data repeating condition checking process (S61). The connection request P1 is the first request received by the repeater 2c and the data repeating condition is in the initial condition. Therefore, user identification process is started (S64) by referring to the user information field P15 stored in the connection request packet.

When irregularity is not detected as the result of user identification, the allowable accessible range of user and matching between the terminal unit in the transmitting side and destination terminal unit is checked (S65). When the allowable accessible range is satisfied, the connection confirming packet P2 setting the normal code "repeating of connection is possible" in the code field is transferred to the transmitting side terminal unit 3b (S66) to start the data transfer condition (S67).

Next, when the connection request packet P1 is received (S60), since the data transfer operation (data repeating) is performed at step S61 for checking the condition, the connection request packet P1 is judged to be received and the repeater is determined (S62) to transfer the connection request packet P1 (S63) by referring to the repeating route control table. At step S64, if irregularity is detected as the result of user identification, the connection confirming packet P2 setting the error code "irregularity of user identification" in the code field P23 is transmitted to the terminal unit 3b which has transmitted the connection request packet P1 (S70) to complete the communication (S71).

At step S65, when the request is out of the accessible range, the connection confirming packet P2 setting the error code "out of the accessible range" in the code field P23 is transmitted to the terminal unit 3b which has transmitted the connection request packet P1 (S68) to complete the communication (S69).

Next, outline of user identification performed in the communication procedures will be explained with reference to FIG. 14 and FIG. 15. In this embodiment, a password identification method will be explained. Various identification methods such as the identification mechanism using a public key and individual identification mechanism have been proposed and this embodiment can be applied to any type of identification mechanism.

FIG. 14 shows a table storing an identification information for utilizing each repeater held by a user 1. The user-held identification information table 400 is constituted by a repeater name 401 in which the repeater name is described and an identification information 402 in which a password information required for identification in each repeater is described. In this example, a user (user 1) is capable of using only the repeater 2a and it has a password "test". When a user (user 1) makes communication via the repeater 2a, it is requested to set this identification information in the user information field P15 of the connection request packet P1.

FIG. 15 shows a table 410 storing the user identification information held by the repeater 2a. In the repeater-held identification information table 410, a user name 411 and a password information 412 of each user are described. In this example, the password of user (user 1) is set to "test", password of user (user 2) to "abcdx", the password of user (user 3) to "poisd" and the password of user (user 4) to "odksci". In this case, if the identification information described in the table is stored in the user information field P15 of the connection request packet P1 when an user 1 to 4 attempts communication via the repeater 2a, such user is identified as the user himself (S22, S64) and the next access control is started (S23, S65).

Next, outline of the access control, to be executed in a company organization as an example, in the communication sequence will be explained with reference to FIG. 16 to FIG. 20.

FIG. 16 shows a table 420 storing user access control information, which are user attributes, held by the repeater 2a. In the user access control table 420 held by the repeater, user name 421 of each user, department 422 to which user belongs, official position of user 423, transmitting side network 424 to which a user can make access, destination network 425 to which a user can make access and services 426 which a user can receive are respectively described.

In this example, a user (user 1) can make access to the network 1a or network 1b from the network 1a or network 1b and the service which a user (user 1) can receive is only the file transfer. A user (user 2) can make access to the

network 1c or network 1e from the network 1c or network 1e and a user (user 2) can receive any kinds of services because "*" is indicated in the service column 426. A user (user 3) can make access to any network from any network because "*" is indicated in the transmitting side column 424 and destination column 425 and can receive the virtual terminal service. A user (user 4) can make access to any network from any network and can receive any services because "*" is indicated in the transmitting side column 424, destination column 425 and service column 426. The asterisk mark "*" indicated in the table means the accessible networks and receivable services. The sign "." means that the item given this mark is not available. As explained above, the regions on the network which a user can use are defined in the transmitting side column 424, destination column 425 and service column 426.

FIG. 17 shows a table 430 storing an access control information of department, which are also user attributes, held in the repeater 2a. The access control table 430 of department held in the repeater describes, for each department, department name 431, accessible destination network 432, accessible transmitting side network 433 and available service 434. In this example, the department "Planning" is capable of making access to the networks 1b, 1c, 1d and 1e from the networks 1b or 1d and can receive only the virtual terminal service. Namely, the regions on the network which each department can use are defined in the destination column 432, transmitting side column 433 and service column 434. As explained, the regions on the network can be defined not only for users but also for one attribute. The asterisk mark "*" described in the table means the accessible network and receivable services. The sign "." means that the item given this mark is not available.

FIG. 18 shows the accessible regions which can be formed depending on the access control information of department. This figure shows the accessible regions of department defined by each table explained above. The accessible region 40a of the Department of General Affairs is the network 1a and network 1b, while the accessible region 40c of the Department of Development and Design is the network 1b, network 1c and network 1e, and the accessible region 40b of the Department of Planning is the network 1b, network 1c, network 1d and network 1e.

As explained above, in this embodiment, the accessible terminal units and application region such as network can be defined for each user depending on the various attributes held by user and moreover the accessible region can also be defined for attribute. As explained, the application regions constituted on the network can form the logical networks for each user, each department and each official position.

FIG. 19 shows the accessible regions when structure of the departments are hierarchically indicated. In this example, the Department of General Affairs 51b of factory A connected to the network 52b of factory A and the Department of General Affairs 51c of factory A connected to the network 52c of factory B can form the accessible region 53 which enables the same work, namely the logical network by defining the Department of General Affairs of factory A as a user or an attribute value of department. The Department of General Affairs 51d of factory B connected to the network 52c of factory B and the Department of General Affairs 51a of laboratory connected to the laboratory network 52a can form, by limiting the services, the region having the properties different from that of the available region 53, namely the available region 54, that is, the logical network which can perform the same work in the Department of General Affairs 51b, 51c of factory A, the Depart-

ment of General Affairs 51d of factory B and the Department of General Affairs 51a of laboratory because the service used for mutual information exchange between the Department of General Affairs 51b, 51c of factory A is fixed to the particular services.

By forming individual networks in different attribute values and properties, the network satisfying individual access policy and security policy can be constituted while offering the transparent network environment.

FIG. 20 shows a table 440 storing access control information of official position, which are also user attributes, held in the repeater 2a. The access control table 440 of official position held in the repeater describes, for each official position name 441, class of transmitting and destination networks 442 indicating the accessible network range, remote destination 443 indicating the accessible destination network and available services 444. The class of transmitting and destination networks 442 indicates the accessible network range. Description "local" indicates that only the network connected to the terminal unit in the transmitting side may be used, while "remote" indicates that the networks other than that connected to the terminal unit in the transmitting side can also be used. The remote destination 443 is effective only when "remote" is set in the transmitting and destination networks 442 and indicates the accessible destination network. In this example, the official position "General Manager" can make access to the network connected to the terminal unit of the transmitting side and to the network other than that connected to the terminal unit in the transmitting side and can make access to any network and receives all services. The asterisk mark "*" described in the table means access to any network is possible and any service can be received. The sign "-" means that the item given this mark is not available.

Relationship between the user access control table 420, department access control table 430 and position access control table 440 will be explained. A user (user 1) belongs to the Department of General Affairs and has the official position "General Manager". A user (user 1) can make access to the network 1a and network 1b and receive the service of only file transfer from the item 427a of user (user 1) in the user access control table 420. Next, from the item 431 of the Department of General Affairs in the department access control table 430, a user (user 1) can make access to the network 1a, network 1b and receive the service of only database access. Moreover, from the item 445a of position "General Manager" in the position access control table 440, the local and remote networks can be used and there is no limitation on the available services.

The access control mechanism solves mismatching of these access control with any one of a rule of logical sum, a rule of logical product and a rule of attribute priority. For instance, in the case of the rule of logical sum, a user (user 1) can make access to the network 1a, network 1b from the network 1a, network 1b and can receive the services of file transfer and database access. In the case of the rule of logical sum, the asterisk mark "*" is excluded from the object. In the case of the logical product, a user (user 1) can make access to the network 1a, network 1b from the network 1a and network 1b but actually can make access within the network 1a and network 1b because there is no receivable service. Moreover, in the case of the rule of attribute priority, the network (Net-1) 1a and network 1b can be used the only the file transfer service can be received by judging the conditions only from user.

A user (user 2) has the official position "Section Chief". In this case, department access control is excluded from the

control object. In the case of the rule by logical sum, a user (user 2) can make access to the network 1c and network 1e from the network 1c and network 1e and receive only the virtual terminal service. Also, in the case of the logical product, a user (user 2) can make access to the network 1c and network 1e from the network 1c and network 1e and receive only the virtual terminal service.

A user (user 3) belongs to the Department of Planning and has the official position "General Manager". In the case of the rule by logical sum, a user (user 3) can make access to the network 1b, network 1c, network (Net-4) 1d and network 1e from the network 1b and network 1d and receive only the virtual terminal service. Also, in the case of the logical product, a user (user 3) can make access to the network 1b, network 1c, network 1d and network (Net-5) 1e from the network 1b, network 1d and can receive only the virtual terminal service.

A user (user 4) belongs to the Department of Planning and does not have any official position. In the case of the rule by logical sum, a user (user 4) can make access to the network 1b, network 1c, network 1d and network 1e from the network 1b and network 1d and can receive only the virtual terminal service. In the case of the rule by logical product, a user (user 4) can make access only in the network 1b and network 1d.

As explained above, the user in the user attribute table 420, 430 or 440 can be defined as not only an individual but also a section, a group or a position.

Next, outline of the data repeating control executed in the communication procedures will be explained with reference to FIG. 21.

FIG. 21 shows the repeating route control table 450 storing the data repeating route information held in the terminal unit 3b in the network 2 and the repeating route control table 451 storing the data repeating route information held in the terminal unit 2c. The tables 450, 451 storing the data repeating route information respectively have a network name describing field 4501 for designating the network which requires repeating and a repeater name describing field 4502 for designating a repeater used for repeating to the network.

The network name describing field 4501 can use a negative operator "-" for description of the part other than the network name described. For instance, "network 2" indicates a "network other than the network 2". In the table 450, a record 4503 indicating "repeating to the network 1 is performed by the repeater 2a", a record 4504 indicating "repeating to the network 3 is performed by the repeater 2b" and a record 4505 indicating "repeating to the network other than the network 2 is performed by the repeater 2c" are registered respectively.

It is also possible to set that repeating to the network 4 and network 5 can be performed by the repeater 2c by sequentially evaluating these records from the record registered previously. In the same manner, in the table 451, a record 4511 indicating "repeating to the network 1 is performed by the repeater 2a", a record 4512 indicating "repeating to the network 3 is performed by the repeater 2b" and a record 4505 indicating "repeating to the network 5 is performed by the repeater 2c" are registered. Description of network and repeater in the table can be realized by designation with a domain name and a host name in DNS or by designation with IP address and net mask.

In above embodiment, various attributes of user, access control information and user identification information are defined for each repeater and each apparatus for making communication. Registration and renewal of these pieces of

information can be executed for each unit from an administration terminal or by using a control unit for simultaneously controlling the repeaters and terminal units for communication.

Moreover, it is also possible to obtain the information by issuing an inquiry at the time of identifying a user and confirming contents of access control by previously registering various attributes of user, access control information and user identification information to information server, etc. such as directory server.

The basis virtual network system and apparatus of this system are explained above but erroneous connection can be prevented by executing mutual identification of terminal unit and repeater when the connection request (S10, S50) in the terminal unit control flowchart and the connection request (S26) in the repeater control flowchart are issued.

FIG. 22 shows an example of the mutual identification method in the communication procedure 1. The identification information table 460 of the terminal unit 3b has an entry 4601 including ID of repeater 2c and a common key 463. The identification information table 461 of the repeater 2c has an entry 4611 including ID of terminal unit 3b and a common key 463 and an entry 4612 including ID of repeater 2d and a common key 464. The identification information table 462 of repeater 2d has an entry 4621 including ID of repeater 2c and a common key 464.

Utilization of the ISO/IEC9798, for example, using the common key explained above realizes mutual identification between the terminal unit 3b and repeater 2c and between the repeater 2c and repeater 2d. The communication data between adjacent apparatuses can also be encrypted depending on the information used in common through the identification process.

FIG. 23 shows an example of the mutual identification system in the communication procedure 2. The identification information table 465 of terminal unit 3b has an entry 4651 including ID of repeater 2c and a common key 468 and an entry 4652 including ID of repeater 2d and a common key 469. The identification information table 466 of repeater 2c has an entry 4661 including ID of terminal unit 3b and a common key 468. The identification information table 467 of repeater 2d has an entry 4671 including ID of terminal unit 3b and a common key 468. Utilization of the common key realizes mutual identification between the terminal unit 3b and repeater 2c and mutual identification between the terminal unit 3b and repeater 2d. Moreover, the communication data between the terminal unit 3b and the repeater 2d adjacent to the terminal unit 3e can also be encrypted depending on the information used in common through the identification process.

When a plurality of repeaters which enable repeating operation to the network exist as shown in FIG. 24, each repeater transmits, to the other repeater or terminal unit, the information of the network through which each repeater can repeat the data and the repeater or terminal unit can realize dynamic selection of route by writing the information received from the other repeater into the table 450 storing the route information.

Moreover, dynamic route selection based on the priority can also be realized by adding the field 4506 indicating priority to the table 450 storing the route information as explained below.

For example, when communication is made between the terminal unit 3b and the terminal unit 3a, the repeaters 2a, 2c become the candidate repeaters for repeating operation. The repeaters 2a, 2c periodically transmit the numerical value information indicating the loading conditions thereof,

the priority field 4506 of records 4507, 4508 in the repeating route information storing table 450 are updated depending on the loading conditions of these repeaters, and the repeaters having higher priority are connected sequentially by referring to the field on the occasion of starting the communication. If connection is rejected, the repeater of the next priority is connected to realize dynamic route selection.

FIG. 25 is a diagram for explaining an example of the communication infrastructure converting function in the virtual network structuring method and apparatus of this system. In this figure, 1101 designates a client computer; 1102, a fire wall and repeating server; 1111, a communication client program; 1121, a data repeating control program; 1103, a server computer; 1131, a server program; 1104, a communication module corresponding to IP V4; 1105, a communication module corresponding to IP V6; 1106, an IP V4 network; 1107, an IP V6 network. The client computer 1101 makes communication conforming to IP V4 protocol using the communication module 1104 corresponding to IP V4. Moreover, the server computer 1103 makes communication conforming to IP V6 protocol using the communication module 1105 corresponding to IP V4.

Therefore, the client computer 1101 and server computer 1103 cannot realize the direct communication. However, the communication between these client computer 1101 and server computer 1103 can be realized by utilizing the data repeating control program 1121 in the fire wall and repeating server 1102 having the IP V4 communication module 1104 and IP V6 communication module 1105. In FIG. 25, conversion between IP V4 and IP V6 has been conducted as an example of the communication infrastructure, but the existing communication infrastructure can also be used by utilizing appropriate repeating program and repeating route table.

FIG. 26 shows a table storing user application log obtained in the repeater. In the user application log table 470, a user name 471, a transmitting side terminal unit 472 used, a destination terminal unit 473 used, a service 474 which a user has received, condition 475 indicating start and end of service, accessibility 476 indicating that connection is accepted in the repeater in which log is collected and time 477 indicating start and end of service are described.

As explained previously, the present invention assures the effect of offering a large scale network system for realizing communication having passed a fire wall by providing a means for exchanging the repeating route information between a plurality of fire walls (repeaters) and of offering a network system having higher security and operation flexibility by realizing access control based on computer users and applications.

Although preferred embodiments of the present invention have been described and illustrated, it will be apparent to those skilled in the art that various modifications may be made without departing from the principles of the invention.

We claim:

1. A repeater for connecting two networks respectively connected to at least one terminal, comprising:

means for receiving a connection request packet designating a destination terminal from a transmission terminal;

means for identifying a user by referring to a user information field stored in said connection request packet;

means for controlling access depending on at least one attribute of said user in said connection request packet, and comprising:

an access control table for storing correspondence between at least one attribute of at least one user and accessible range of said networks; and

means for checking said at least one attribute of said user in said connection request packet with said accessible range of said networks according to said access control tablet;

means for transmitting said connection request packet to a next (stage) repeater provided to identify said user by referring to said user information field stored in said connection request packet;

a repeating route control table for storing at least one correspondence between a first address area designated by excluding specified address area and an address of another device provided to transfer the data to said first address area, and for storing correspondence between a second address area including said destination terminal and an address of another repeater provided to transfer the data to said second address area;

means for making a comparison between the destination terminal name field of said connection request packet and said destination terminal according to said repeating route control table; and

means for making a determination of the next (stage) repeater with reference to said repeating route control table based on said comparison.

2. The repeater according to claim 1, wherein said means for controlling access executes access control by combining said attributes of said user in said access control table according to a predetermined rule.

3. The repeater according to claim 2, wherein said attribute of said user in said access control table comprises at least one of a user name, an official position and a department.

4. The repeater according to claim 3, further comprising: means for changing said attribute of said user in said access control table according to information received via said networks.

5. A repeater for connecting two networks each being connected to at least one terminal, said repeater comprising:

a repeating route control table for storing at least one correspondence between a first address area designated by excluding a specified address area and an address of another repeater provided to transfer the data to said first address area, and for storing correspondence between a second address area including said destination terminal and an address of another repeater provided to transfer the data to said second address area;

means for receiving a connection request packet designating a destination terminal from a transmission terminal;

means for making a comparison between the destination terminal name field of said connection request packet and said destination terminal according to said repeating route control table;

means for making a determination of a next (stage) repeater with reference to said repeating route control table based on said comparison; and

means for transmitting said connection request packet to said next (stage) repeater based on said determination.

6. The repeater according to claim 5, further comprising: means for changing contents of said repeating route control table according to information received via at least one of said networks.

7. The repeater according to claim 6, wherein said means for changing changes said contents depending on a load condition, a fault condition or a designation of application running on said at least one terminal.

8. A computer program stored on a storage medium, for repeating a communication, when said computer program is executed by a computer which connects two networks each being connected to at least one terminal, said computer program causes said computer to perform the steps of:

receiving a connection request packet designating a destination terminal of said at least one terminal from a transmission terminal of said at least one terminal;

identifying a user by referring to a user information field stored in said connection request packet;

controlling access depending on at least one attribute of said user in said connection request packet according to an access control table which stores correspondence between at least one attribute of at least one user and accessible range of said networks; and

transmitting said connection request packet to a next (stage) repeater provided to identify said user by referring to said user information field stored in said connection request packet.

9. The computer program according to claim 8, wherein said controlling access step includes checking said at least one attribute of said user in said connection request packet with said accessible range of said networks according to said access control table.

10. The computer program according to claim 9, further causing said computer to perform the steps of:

making a comparison between the destination terminal name field of said connection request packet and the repeater name according to a repeating route control table which stores correspondence between an address area including said destination terminal and an address of another repeater provided to transfer the data to said address area; and

determining a next (stage) repeater with reference to said repeating route control table based on said comparison.

11. The computer program according to claim 9, wherein said controlling access step further includes combining said attributes of said user in said access control table according to a predetermined rule.

12. The computer program according to claim 11, wherein said attribute of said user in said access control table comprises at least one of a user name, an official position and a department.

13. The computer program according to claim 12, further causing said computer to perform the step of:

changing said attribute of said user in said access control table according to information received via said networks.

14. A computer program stored on a storage medium, for repeating a communication, when said computer program is executed by a computer which connects two networks each being connected to at least one terminal, said computer program causes said computer to perform the steps of:

receiving a connection request packet designating a destination terminal of said at least one terminal from a transmission terminal of said at least one terminal;

making a comparison between the destination terminal name field of said connection request packet and a repeater name according to a repeating route control table which stores correspondence between an address of said destination terminal and an address of another repeater provided to transfer the data to the address;

making a determination of a next (stage) repeater with reference to said repeating route control table based on said comparison; and

17

transmitting a connection confirming packet to said destination terminal, a packet making said terminal transmit another connection request packet based on said determination.

15. The computer program according to claim 14, further causing said computer to perform the step of:

changing contents of said repeating route control table according to information received via at least one of said networks.

16. The computer program according to claim 15, wherein said changing step further includes changing said contents depending on a load condition, a fault condition or a designation of application running on said at least one terminal.

17. The computer program according to claim 16, wherein said storage medium is included in a server connected to said at least one terminal of said networks; and wherein said server transfers said computer program stored on said storage medium to said computer connected to said at least one terminal of said networks.

18. A computer program according to claim 14 wherein said connection confirming packet includes a code making the transmission terminal judge whether the connection to said destination terminal is completed or not.

19. A method for connecting two networks each being connected to at least one terminal, comprising the steps of:

receiving a connection request packet designating a destination terminal from a transmission terminal;

identifying a user by referring to a user information field stored in said connection request packet;

controlling access depending on at least one attribute of said user in said connection request packet according to an access control table which stores correspondence between at least one attribute of at least one user and accessible range of said networks; and

transmitting said connection request packet to a next (stage) repeater provided to identify said user by referring to said user information field stored in said connection request packet.

20. The method according to claim 19, wherein said controlling access step includes checking said at least one attribute of said user in said connection request packet with said accessible range of said networks according to said access control table.

21. The method according to claim 20, further causing said computer to perform the steps of:

making a comparison between the destination terminal name field of said connection request packet and the repeater name according to a repeating route control table which stores correspondence between an address area including said destination terminal and an address of another repeater provided to transfer the data to said address area; and

making a determination of the next (stage) repeater with reference to said repeating route control table based on said comparison.

22. The method according to claim 20, wherein said controlling access step further includes the sub step of:

combining said attributes of said user in said access control table according to a predetermined rule.

23. The method according to claim 22, wherein said attribute of said user in said access control table comprises at least one of a user name, an official position and a department.

18

24. The method according to claim 23, further causing said computer to perform the step of:

changing said attribute of said user in said access control table according to information received via said networks.

25. A method for connecting two networks each being connected to at least one terminal, comprising the steps of:

receiving a connection request packet designating a destination terminal from a transmission terminal;

making a comparison between the destination terminal name field of said connection request packet and a repeater name according to a repeating route control table which stores correspondence between an address of said destination terminal and an address of another repeater provided to transfer the data to the address;

making a determination of a next (stage) repeater with reference to said repeating route control table based on said comparison; and

transmitting said connection request packet to said next (stage) repeater based on said determination.

26. The method according to claim 25, further causing said computer to perform the step of:

changing contents of said repeating route control table according to information received via at least one of said networks.

27. The method according to claim 26, wherein said changing step further includes changing said contents depending on a load condition, a fault condition of said repeater or a designation of application running on said at least one terminal.

28. A network system having at least two networks each being connected to at least one terminal, said network system comprising:

a transmission terminal for transmitting a connection request packet designating a destination terminal and including at least one user attribute in a user information field;

a repeater for connecting said networks to each other, said repeater comprising means for receiving said connection request packet, and means for identifying said user by referring to said user information field stored in said connection request packet;

a destination terminal for transmitting a connection confirming packet as a response to said connection request packet, said destination terminal comprising: means for receiving said connection request packet, and means for identifying said user by referring to said user information field stored in said connection request packet,

said transmission terminal confirming that each of said repeater and said destination terminal identifies said user and a communication route between said transmission terminal and said destination terminal is established.

29. The network system according to claim 28, further comprising:

means for controlling access depending on at least one attribute of said user in said connection request packet, wherein said means for controlling access comprises: an access control table for storing correspondence between at least one attribute of at least one user and accessible range of said networks; and means for checking said at least one attribute of said user in said connection request packet with said accessible range of said networks according to said access control table.

19

30. The network system according to claim 29, wherein said repeater further comprises:

a repeating route control table for storing at least one correspondence between a first address area designated by excluding a specified address area and an address of another repeater provided to transfer the data to said first address area, and for storing correspondence between a second address area including said destination terminal and an address of another repeater provided to transfer the data to said second address area;

means for making a comparison between the destination terminal name field of said connection request packet and the destination terminal name according to said repeating route control table; and

20

means for determining a next (stage) repeater with reference to said repeating route control table based on said comparison.

31. The network system according to claim 28, wherein said repeater further comprises means for transmitting said connection request packet to the next (stage) repeater based on access control information, said next (stage) repeater provided to identify said user referring to said user information field stored in said connection request packet.

32. The network system according to claim 28, wherein said repeater further comprises means for transmitting said connection confirming packet to said transmission terminal based on access control information.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO : 6,111,883

DATED : August 29, 2000

Page 1 of 2

INVENTOR(S) : Masato TERADA et al., et al.

It is certified that error appears in the above-identified patent and that said Letters Patent are hereby corrected as shown below:

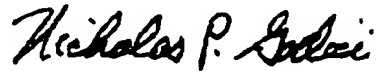
On title page, item 75 Inventors, replace "Tetsuya" with --Tatsuya--.

Column 12, line 31, replace "FIG. 21" with -- FIGs. 21A-21B --;

Column 12, line 32, replace "FIG. 21" with -- FIGs. 21A-21B --; and
Figure 21, please replace with Figures 21(A) and 21(B) as follows:

Signed and Sealed this
Eighth Day of May, 2001

Attest:



NICHOLAS P. GODICI

Attesting Officer

Acting Director of the United States Patent and Trademark Office



US005832228A

United States Patent [19][11] Patent Number: **5,832,228****Holden et al.**[45] Date of Patent: **Nov. 3, 1998**

[54] **SYSTEM AND METHOD FOR PROVIDING MULTI-LEVEL SECURITY IN COMPUTER DEVICES UTILIZED WITH NON-SECURE NETWORKS**

Primary Examiner—Parshotam S. Lall
Assistant Examiner—Viet Vu
Attorney, Agent, or Firm—Plevy & Associates

[57] **ABSTRACT**

A multi-level network security system is disclosed for a computer host device coupled to at least one computer network. The system including a secure network interface Unit (SNIU) contained within a communications stack of the computer device that operates at a user layer communications protocol. The SNIU communicates with other like SNIU devices on the network by establishing an association, thereby creating a global security perimeter for end-to-end communications and wherein the network may be individually secure or non-secure without compromising security of communications within the global security perimeter. The SNIU includes a host/network interface for receiving messages sent between the computer device and network. The interface operative to convert the received messages to and from a format utilized by the network. A message parser for determining whether the association already exists with another SNIU device. A session manager coupled to said network interface for identifying and verifying the computer device requesting access to said network. The session manager also for transmitting messages received from the computer device when the message parser determines the association already exists. An association manager coupled to the host/network interface for establishing an association with other like SNIU devices when the message parser determines the association does not exist.

[75] Inventors: James M. Holden, Valley Center; Stephen E. Levin, Poway, both of Calif.; James O. Nickel, Dayton, Md.; Edwin H. Wrench, San Diego, Calif.

[73] Assignee: ITT Industries, Inc., White Plains, N.Y.

[21] Appl. No.: 688,543

[22] Filed: Jul. 30, 1996

[51] Int. Cl.⁶ G06F 13/00

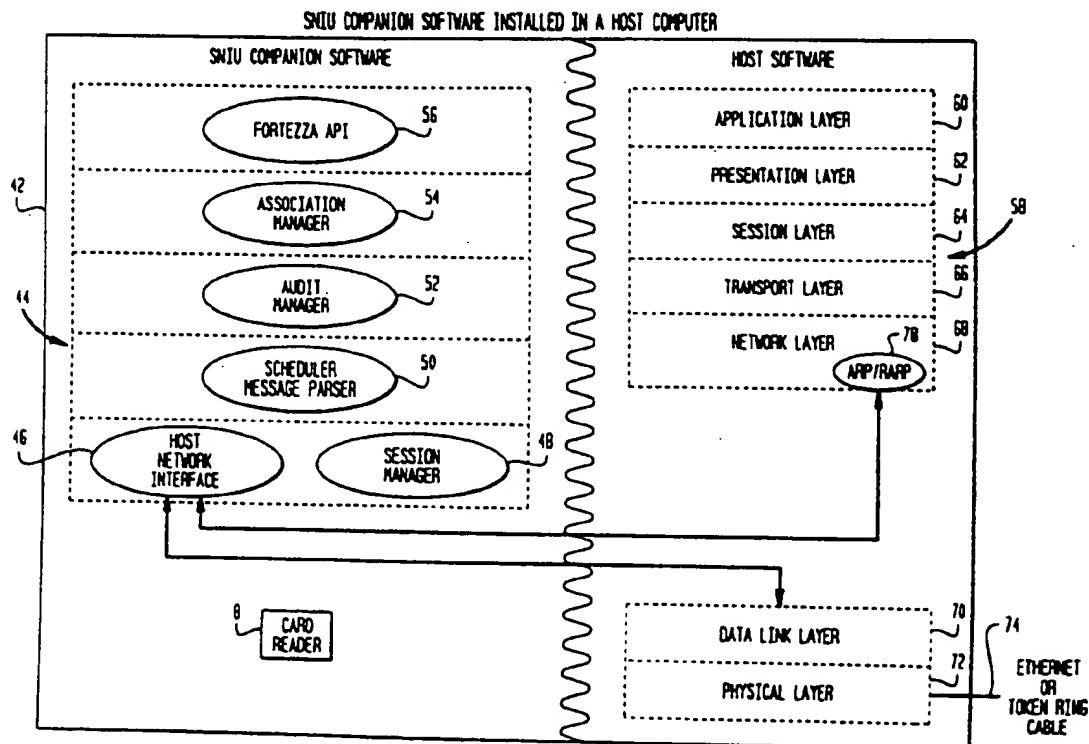
[52] U.S. Cl. 395/200.55; 395/200.59; 395/200.8; 395/187.01

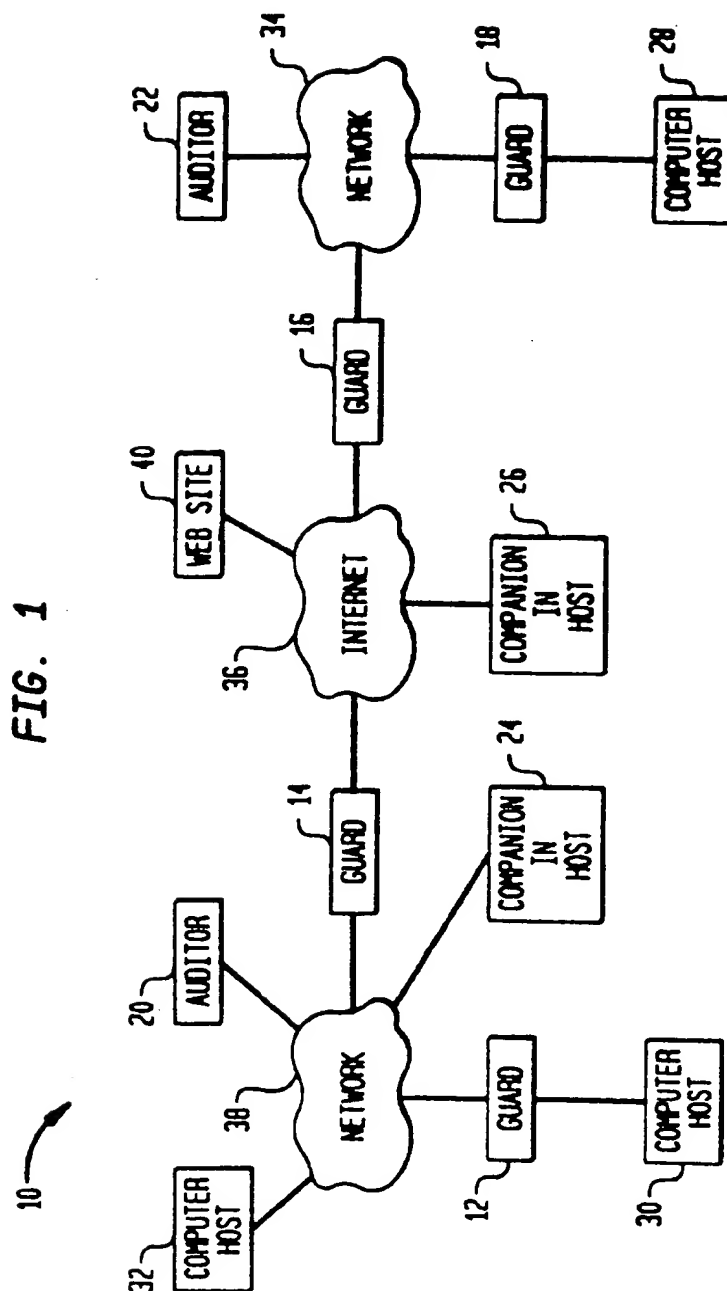
[58] Field of Search 395/200.43, 200.46, 395/200.47, 200.49, 200.53, 200.55, 200.57, 200.59, 200.6, 200.8, 187.01; 380/4, 25

[56] **References Cited****U.S. PATENT DOCUMENTS**

4,885,789	12/1989	Burger et al.	380/25
5,204,961	4/1993	Barlow	395/187.01
5,361,359	11/1994	Tajalli et al.	395/186
5,416,842	5/1995	Aziz	380/30
5,577,209	11/1996	Boyle et al.	395/187.01
5,619,657	4/1997	Sudama et al.	395/200.55

20 Claims, 4 Drawing Sheets





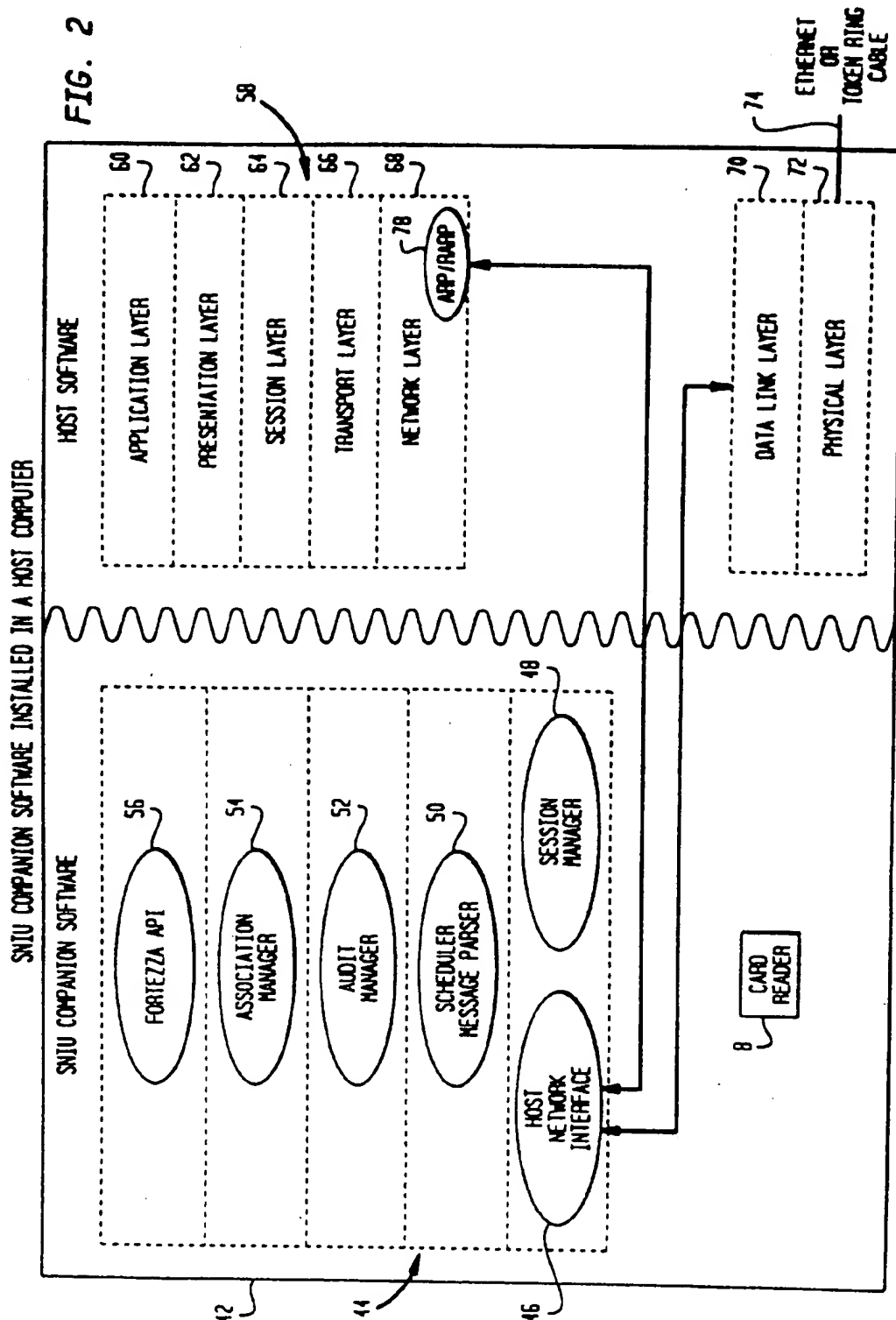


FIG. 3

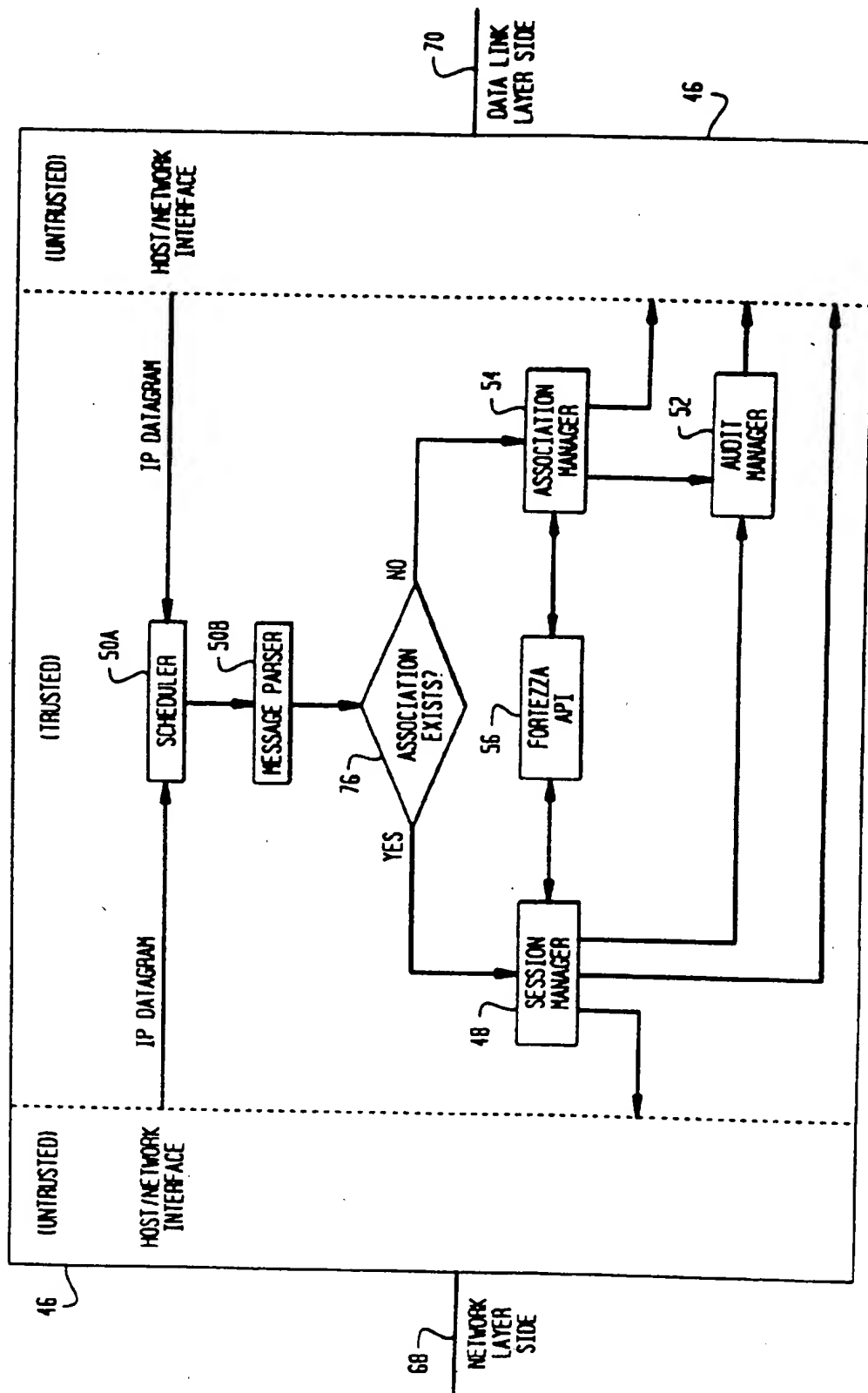


FIG. 4

0	8	16	24	31
HARDWARE TYPE		PROTOCOL TYPE		
HLN	PLEN	OPERATION		
SENDER'S HA (BYTES 0-3)				
SENDER'S HA (BYTES 4-5)		SENDER'S IP (BYTES 0-1)		
SENDER'S IP (BYTES 2-3)		TARGET'S HA (BYTES 0-1)		
TARGET'S HA (BYTES 2-3)				
TARGET'S IP (BYTES 0-4)				

FIG. 5

0	8	16	24	31
NEXT				
PREVIOUS				
IP ADDRESS				
PEER SNTU IP ADDRESS				
ASSOCIATION KEY POINTER				
RELEASE KEY POINTER				
ASSOC. TYPE		RELKEY. TYPE		SECURITY. LEVEL
				SPARE

FIG. 6

0	8	16	24	31
NEXT				
PREVIOUS				
DISTINGUISHED NAME (BYTES 0-3)				
...				
DISTINGUISHED NAME (BYTES 28-31)				
MEK (BYTES 0-3)				
MEK (BYTES 4-7)				
MEK (BYTES 8-11)				
IV (BYTES 0-3)				
IV (BYTES 4-7)				
IV (BYTES 8-11)				
IV (BYTES 12-15)				
IV (BYTES 16-19)				
IV (BYTES 20-23)				
CERTIFICATE POINTER				
INDEX		SPARE		SPARE

SYSTEM AND METHOD FOR PROVIDING MULTI-LEVEL SECURITY IN COMPUTER DEVICES UTILIZED WITH NON-SECURE NETWORKS

RELATED APPLICATIONS

The Assignee herein, ITT Corporation, is the record owner of co-pending U.S. application Ser. No. 08/270,398 to Boyle et al., entitled APPARATUS AND METHOD FOR PROVIDING NETWORK SECURITY, filed Jul. 5, 1994 now U.S. Pat. No. 5,577,209

FIELD OF THE INVENTION

The present invention relates in general to secure and multi-level secure (MLS) networks and in particular to a system and method for providing security and multi-level security for computer devices utilized in non-secure networks.

BACKGROUND OF THE INVENTION

Multi-level secure (MLS) networks provide a means of transmitting data of different classification levels (i.e. Unclassified, Confidential, Secret and Top Secret) over the same physical network. To be secure, the network must provide the following security functions: data integrity protection, separation of data types, access control, authentication and user identification and accountability.

Data integrity protection ensures that data sent to a terminal is not modified enroute. Header information and security level are also protected against uninvited modification. Data integrity protection can be performed by check sum routines or through transformation of data, which includes private key encryption and public key encryption.

Separation of data types controls the ability of a user to send or receive certain types of data. Data types can include voice, video, E-Mail, etc. For instance, a host might not be able to handle video data, and, therefore, the separation function would prevent the host from receiving video data.

Access control restricts communication to and from a host. In rule based access control, access is determined by the system assigned security attributes. For instance, only a user having Secret or Top Secret security clearance might be allowed access to classified information. In identity based access control, access is determined by user-defined attributes. For instance, access may be denied if the user is not identified as an authorized participant on a particular project. For control of network assets, a user may be denied access to certain elements of the network. For instance, a user might be denied access to a modem, or to a data link, or to communication on a path from one address to another address.

Identification of a user can be accomplished by a unique name, password, retina scan, smart card or even a key for the host. Accountability ensures that a specific user is accountable for particular actions. Once a user establishes a network connection, it may be desirable that the user's activities be audited such that a "trail" is created. If the user's actions do not conform to a set of norms, the connection may be terminated.

Currently, there are three general approaches to providing security for a network: trusted networks, trusted hosts with trusted protocols, and encryption devices. The trusted network provides security by placing security measures within the configuration of the network. In general, the trusted network requires that existing protocols and, in some cases,

physical elements be replaced with secure systems. In the Boeing MLS Lan, for instance, the backbone cabling is replaced by optical fiber and all access to the backbone is mediated by security devices. In the Verdix VSLAN, similar security devices are used to interface to the network, and the network uses encryption instead of fiber optics to protect the security of information transmitted between devices. VSLAN is limited to users on a local area network (LAN) as is the Boeing MLS Lan.

Trusted hosts are host computers that provide security for a network by reviewing and controlling the transmission of all data on the network. For example, the U.S. National Security Agency (NSA) has initiated a program called Secure Data Network System (SDNS) which seeks to implement a secure protocol for trusted hosts. In order to implement this approach, the installed base of existing host computers must be upgraded to run the secure protocol. Such systems operate at the Network or Transport Layers (Layers 3 or 4) of the Open Systems Interconnection (OSI) model.

Encryption devices are used in a network environment to protect the confidentiality of information. They may also be used for separation of data types or classification levels. Packet encryptors or end-to-end encryption (EEE) devices, for instance, utilize different keys and labels in protocol headers to assure the protection of data. However, these protocols lack user accountability since they do not identify which user of the host is using the network, nor are they capable of preventing certain users from accessing the network. EEE devices typically operate at the Network Layer (Layer 3) of the OSI model. There is a government effort to develop cryptographic protocols which operate at other protocol layers.

An area of growing concern in network security is the use of computer devices in non-secure networks. Such computer devices often include valuable information, which may be lost or stolen due to these computers being accessed through the non-secured network. In light of this problem, a number of related products have been developed. The products developed include Raptor Eagle, Raptor Remote, Entrust, Secret Agent and Veil. Although, these products serve the same purpose, a number of different approaches have been utilized. For example, Raptor Eagle, Raptor Remote, and Veil implement these products as software instantiations. While Entrust and Secret Agent utilize hardware cryptographic components. Additionally, Raptor products are also application independent.

A problem with the above described products is that none are based upon the use of highly trusted software. Veil is an off-line encryption utility, which cannot prevent the inadvertent release of non-encrypted information. While Raptor Eagle and Raptor Remote are based on software instantiations and thus cannot be verified at the same level of assurance. Secret Agent and Entrust while hardware based are dependent upon the development of integration software for specific applications.

It is therefore, an objective of the present invention, to provide a multi-level security system that is readily adaptable to computer devices which provides an adequate level of security assurances.

SUMMARY OF THE INVENTION

In accordance with the present invention, a network security apparatus and method for a network comprises a secure network interface unit (SNIU) coupled between host computer or user computer unit, which may be non-secure,

and a network (i.e. a SNIU can be placed between two networks), which may be non-secure. When an SNIU is implemented at each computer unit to be secured on the network, a global security perimeter is provided for ensuring security policy enforcement, controlled communication release, controlled communication flow, and secure session protocols through each computer unit interface.

In a preferred embodiment, the SNIU is configured to process a defined trusted session protocol (TSP) and perform the core functions of host/network interface by utilizing an association manager, session manager and data sealer. The user/service interface function performs a standard communications stack function by handling all of the standard communications data translation between the Physical Data Link and Network protocol layers (i.e. layers one thru three). The host/network interface does not require the same level of trust as the rest of SNIU's software. This allows this software to be logically and physically separated from the rest of the software without effecting the underlying security of the system as a whole. The association manager functions include host computer and peer SNIU identification, audit, association setup and termination and maintenance of the sealer keys generated for the association between the two peer SNIUs. The session manager functions include sealing, verifying message authentication codes, audit and enforcing a security on each datagram passed thru the SNIU.

A software SNIU is also disclosed contained within a communications stack of a portable computer device operating at a user layer communications protocol. The software SNIU contains the association and session managers as previously described, but not a host/network interface as the function is performed by the communications stack of the host computer. The SNIU is capable of communicating with other like SNIU devices creating a global security perimeter for end-to-end communications and wherein the network may be individually secure or non-secure without compromising security of communications within the global security perimeter.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic diagram of an MLS network system in accordance with the present invention;

FIG. 2 is a block diagram of the software SNIU installed in a computer host in accordance with the present invention;

FIG. 3 is a data flow diagram for the software SNIU in accordance with the present invention;

FIG. 4 is a table showing the format for the Address Resolution Protocol and Reverse Address Resolution Protocol messages in accordance with the present invention;

FIG. 5 is a table showing the data structure for a token of the Association Table in accordance with the present invention; and

FIG. 6 is a table showing the data structure of a token for the Sym_Key Table in accordance with the present invention.

DETAILED DESCRIPTION

The present invention is directed to a secure network interface unit (SNIU), which is utilized to control communications between a user such as a computer host and a network at a "session layer" of interconnection which occurs when a user on the network is identified and a communication session is to commence. For example, the industry-standard Open Systems Interconnection (OSI) model, defines seven layers of a network connection: (1) physical;

(2) data link; (3) network; (4) transport; (5) session; (6) presentation; and (7) application. In the present invention, the network security measures are implemented at the Session Layer 5. The placement of security at the Session Layer allows existing network assets and existing network protocols at the Transport Layer 4 and lower to continue to be used, thereby avoiding the need to replace an installed network base for the implementation of the multi-level security system. The connected host or user equipment and the network backbone are therefore not required to be secure (trusted). Conventionally, OSI network applications employ CCITT X.215 which is a non-secure session layer protocol. None of the prior multi-level security systems employ the security measures described herein in the Session Layer.

The SNIU according to the present invention can be configured in a number of different embodiments depending on the particular physical locations and forms of implementation. These embodiments include a stand alone hardware SNIU and a software SNIU.

The hardware embodiment of the SNIU is implemented solely as a stand alone hardware device. Such a configuration is desirable, since the stand alone SNIU is highly trusted. The stand alone SNIU is configured to be inserted between existing hosts and a network. The SNIU is transparent to the host, and any legacy system or application software running on the host. The stand alone SNIU provides protection for any host connected to an IP based network. There is no requirement that the attached host computers run a trusted operating system. The stand alone SNIU provides a trusted boundary between the protected hosts and the unprotected network. Protected means that the connection is with another known SNIU (a unique digital signature identifies the SNIU), the messages are confidential (encrypted) and unaltered (cryptographic residues validate the packet).

The software embodiment of the SNIU is implemented solely as a software function resident in and executed from the host machine. Such a configuration is desirable, since the software SNIU is designed to be installed in existing portable computers, which avoids the additional cost of additional hardware required by a stand alone hardware SNIU. The software SNIU provides the same network security features as the stand alone SNIU when the host computer is connected to home enterprise's network. The software SNIU also extends that same level of security across the Internet (or any other unprotected network) when the user is on the road and is remotely communicating with the enterprise network or other remotely located computer devices including a similar software SNIU.

The software SNIU provides all of the functionality and security of the stand alone SNIU as well as complete operability with these devices. The software comprising the software SNIU is based on the same software utilized in the stand alone hardware SNIU. The user of the software SNIU assumes an acceptable risk in exchange for not requiring additional hardware required by a stand alone SNIU, which cannot be circumvented or corrupted via attacks from originating from external hardware. By providing reasonable software protection (not allowing unauthorized personal physical access) and software protection (anti-virus protection), a software SNIU can be utilized providing the user with an acceptable level of risk. If the user is confident that the software comprising the software SNIU is not circumvented or modified, then he can enjoy the same degree of confidence as the user of a stand alone device.

Referring to FIG. 1, there is shown an example of a Multi-Level Security (MLS) System in accordance with the

present invention. This system 10 incorporates the various embodiments of the SNIUs in order to provide MLS for computer networks such as the Internet. For example, the guard devices 14,16 which are hardware embodiments of the SNIU are coupled between computer networks 34,36,38 providing inter-network security. Additional guard devices 12,18 are coupled between users such as computer hosts 28,30,32 and the respective networks 30,32,34. The software embodiment of the SNIU are implemented as companions within computer hosts 24,26, which provides network security without requiring additional hardware. The auditors 20,22 are also hardware SNIUs which are configured to communicate directly with the other SNIUs to log audit events and potentially signal alarms. The above described system 10 enables secured and non-secured users such as a web site 40 to communicate with each other without the danger of compromising security.

During operation, the SNIUs included in the above described system 10 communicate with each other thereby creating a global security perimeter for end-to-end communications and wherein the network may be individually secure or non-secure without compromising security of communications within the global security perimeter. The SNIUs are capable of passing digital data, voice and video traffic so as to provide the full functionality required for a Trusted Session Protocol (TSP). The TSP uses the facilities of the lower level protocols to transmit data across the networks. To this end, and to provide flexibility, the specialized network interface SNIU is designed to allow coupling of the TSP with existing (non-secure) equipment and underlying network.

Security System Policies

The SNIU devices in accordance with the present invention may implement a number of security policies suitable to the circumstances of a given network environment. The major policy areas are: discretionary access control; mandatory access control; object reuse; labeling; identification and authentication; audit; denial of service detection; data type integrity; cascading control; and covert channel use detection.

Discretionary access control is a means of restricting access to objects (data files) based on the identity (and need to know) of the user, process, and/or group to which the user belongs. It may be used to control access to user interface ports based on the identity of the user. For a single-user computer unit, this mechanism may be implemented in the SNIU, whereas for a multi-user host, the DAC control may be implemented at the host machine. Discretionary access control may also be implemented as discretionary dialog addressing, wherein the addressing of all communications originated by a user is defined, and for user discretionary access denial, wherein a user may refuse to accept a communication from another user.

Mandatory access control is a means of restricting access to objects based on the sensitivity (as represented by a classification label) of the information contained in the objects, and the formal authorization (i.e., clearance) of the user to access information of such sensitivity. For example, it may be implemented as dialog lattice-based access control, wherein access requires a correct classification level, integrity level, and compartment authorization, dialog data-type access control, wherein correct data type authorization is required for access, and cascade protection, wherein controls are provided to prevent unauthorized access by cascading user access levels in the network.

Object reuse is the reassignment and reuse of a storage medium (e.g., page frame, disk sector, magnetic tape) that

once contained one or more objects to be secured from unauthorized access. To be secured, reused, and assigned to a new subject, storage media must contain no residual data from the object previously contained in the media. Object reuse protection may be implemented by port reuse protection, session reuse protection, dialog reuse protection, and/or association reuse protection.

Labeling requires that each object within the network be labeled as to its current level of operation, classification, or accreditation range. Labeling may be provided in the following ways: user session security labeling, wherein each user session is labeled as to the classification of the information being passed over it; dialog labeling, wherein each dialog is labeled as to the classification and type of the information being passed over it; and host accreditation range, wherein each host with access to the secured network is given an accreditation range, and information passing to or from the host must be labeled within the accreditation range.

Identification is a process that enables recognition of an entity by the system, generally by the use of unique user names. Authentication is a process of verifying the identity of a user, device, or other entity in the network. These processes may be implemented in the following ways: user identification; user authentication; dialog source authentication, wherein the source of all communication paths is authenticated at the receiving SNIU before communication is allowed; SNIU source authentication, wherein the source SNIU is authenticated before data is accepted for delivery; and

administrator authentication, wherein an administrator is authenticated before being allowed access to the Security Manager functions.

An audit trail provides a chronological record of system activities that is sufficient to enable the review of an operation, a procedure, or an event. An audit trail may be implemented via a user session audit, a dialog audit, an association audit, an administrator audit, and/or a variance detection, wherein audit trails are analyzed for variance from normal procedures.

Denial of service is defined as any action or series of actions that prevent any part of a system from functioning in accordance with its intended purpose. This includes any action that causes unauthorized destruction, modification, or delay of service. The detection of a denial of service may be implemented for the following condition: user session automatic termination, such as when unauthorized access has been attempted; user machine denial of service detection, such as detection of a lack of activity on a user machine; dialog denial of service detection; association denial of service detection, such as detection of a lack of activity between SNIUs; and/or data corruption detection, such as when an incorrect acceptance level is exceeded.

Covert channel use is a communications channel that allows two cooperating processes to transfer information in a manner that violates the system's security policies. Detection of covert channel use may be implemented, for example, by delay of service detection, such as monitoring for unusual delays in message reception, or dialog sequence error detection, such as monitoring for message block sequence errors.

Details Of the Software SNIU

Referring to FIG. 2, a block diagram of the software SNIU installed in a computer host is shown. The software SNIU 44 is implemented as a software function within a host computer 42. The SNIU 42 interfaces with the communications stack of the host computer 58 in order to send and receive

messages over the Ethernet or token ring cable 74. The communications stack 58 is a typical OSI model including a physical 72, data link layer 70, network layer 68, transport layer 66, session layer 64, presentation layer 62 and application layer 60. The network layer 68 includes an ARP/RARP module which is utilized to process Address Resolution Protocol (ARP) and Reverse Address Resolution Protocol (RARP).

As can be seen from FIG. 2, the SNIU 44 is installed between the network and data link layers of the communications stack 68,70, which enables it to be transparent to the other high order software.

The main modules of the SNIU include a Host/Network Interface 46, Session Manager 48, Trusted Computing Base 50, Audit Manager 52, Association Manager 54 and Fortezza API 56.

The primary data structures included in the SNIU are the Association Table, Sym_Key Table, Certificate Table, Waiting Queue and Schedule Table. These data structures are described later in the description of the protocol.

The Host/Network Interface 46 provides the interfacing between the SNIU 44 and communications stack 58. The Fortezza API 56 is a driver for the card reader 76 included in the host computer 42. The card reader 76 is adapted to receive a Fortezza card which is a PCMCIA card configured to perform integrity and authenticating functions. The Fortezza card performs the integrity function by encrypting messages leaving the SNIU 44 and decrypting incoming messages. The authentication function is accomplished by the Fortezza card generating and reading digital signatures which are unique to each SNIU. The Fortezza card includes a private key to generate the digital signature and a public key to read the signatures. The other SNIU modules will be described in conjunction with data flow diagram of FIG. 3.

Referring to FIG. 3, there is shown a data flow diagram for the software SNIU. When the host computer communicates with another computer over a network, the communications protocol stack within the computer processes the data to be transmitted. If a user on the computer is transmitting a file to another computer, the user may select the file to send by interacting with application layers software. The display which the user sees is controlled by presentation layer software. Session layer software checks the users permission codes to determine if the user has access to the file. Transport layer software prepares Internet Protocol Datagrams containing blocks of file data and determines that the transmitted file data is properly received and acknowledged or is re-transmitted.

The Host/Network interface 46 is utilized to intercept the data packets transmitted between the network and data link layers 68,70. The interface 46 is utilized to format the data packets into an appropriate format depending on whether the data packet is incoming or out going. The interface 46 accomplishes this by removing the hardware address header when it receives a data packet and re-applies the same header when the packet is released (even if the underlying IP address header was changed). Since the interface in the software SNIU 46 does not handle ARP and RARP message for the host computer, it can be smaller than the one utilized in the hardware SNIU. As previously described, the ARP/RARP module included in the network layer 68 performs this function.

When the untrusted Host/Network Interface 46 completes re-assembling an IP datagram from a host computer, the datagram is passed to the Trusted Computing Base 50 (TCB) of the SNIU for processing. The TCB 50 is the collection of hardware and software which can be trusted to enforce the

security policy. The TCB 50 includes a Scheduler module 50A and a Message Parser 50B. The Scheduler 50A is utilized to control the flow of datagrams within the SNIU. The scheduler 50A provides timing for incoming datagrams and temporarily stores these datagrams in a buffer if earlier ones are being processed.

The Message Parser 50B is the first module in the TCB which processes an IP datagram received from the host computer. The Message Parser 50B checks the Association Table 76 and determines whether or not an association already exists for sending the datagram to its destination. If no association exists, the datagram is stored on the Waiting Queue and the Association Manager 54 is called to establish an association between this SNIU and the SNIU closest to the destination host. If an association does exist, the Session Manager 48 is called to encrypt the datagram, check security rules, and send the encrypted Protected User Datagram (PUD) to the peer SNIU.

When the Association Manager 54 is called, it prepares two messages to initiate the association establishment process. The first message is an Association Request Message which contains the originating host computer level and this SNIU's certificate (containing it's public signature key). This message is passed to the Fortezza API 56 which controls the Fortezza card which signs the message with this SNIU's private signature key. The second message is an ICMP Echo Request message which will be returned to this SNIU if it is received by the destination host. Both messages are passed to the network-side Host/Network Interface Module 46 to be transmitted to the destination host.

When a SNIU receives messages, the messages are first processed by the SNIU's receiving port's Host/Network Interface 46 which reassembles the messages and passes them to the trusted software. The Message Parser module 50B passes the Association Request Message to the Association Manager 54 module and deletes the ICMP Echo Request message. The Association Manager 54 passes the message to the Fortezza API 56 which verifies the digital signature. If not valid, the Audit Manager 52 is called to generate an Audit Event Message to log the error. If the signature is OK, the Association Manager 54 saves a copy of the received Association Request Message in the Waiting Queue, adds this SNIU's certificate to the message, calls the Fortezza API 56 to sign the message, generates a new ICMP Echo Request message, and passes both messages to the Host/Network Interface module 46 to transmit the messages to the destination host. If the messages are received by any other SNIU's before reaching the destination host, this process is repeated by each SNIU.

If the destination host computer does not contain the Companion SNIU software, the host's communications protocol stack software automatically converts the ICMP Echo Request message to an ICMP Echo Reply and returns it to the SNIU which sent it. However, the destination host does not contain any software which can process the Association Request Message; so it is ignored (i.e., deleted).

If the destination host computer does contain Companion SNIU software, the host's data link layer software converts the stream of bits from the physical layer into packets which are passed to the Companion's Host/Network Interface module 46. The hardware address headers are stripped off of the packets and saved; and the packets are re-assembled into IP datagrams which are passed to the Message Parser 50B. The ICMP Echo Request message is ignored; and the Association Request Message is passed to the Fortezza API 56 to have the signature verified. If valid, the message is passed to the Association Manager module 54 which saves

the originating host and SNIU data and generates an Association Grant Message. This message contains the SNIU's IP address (which is the same as the destination host's), the SNIU's certificate, the host's security level, and sealer keys for the originating SNIU and the previous intermediate SNIU (if there was one). The sealer keys (a.k.a. Message Encryption Keys) are explained elsewhere.

The Fortezza API 56 is then called to sign the message which is passed to the Host/Network Interface module 46. The Association Grant Message is converted from an IP datagram to network packets and passed back to the host's hardware packet drivers (in the data link layer) for transmission back to the originating host.

Any intermediate SNIU's which receive the Association Grant Message process the message up through the communications stack protocol layers to the network layer which calls the Message Parser 50B to process the message. The signature on the message is verified by the Fortezza API 56 and audited via the Audit Manager 52 if not valid. Otherwise, the validated message is processed by the Association Manager 54 module which removes and saves one of the sealer keys (a.k.a. a release key) which will be used by this SNIU and the previous SNIU (which generated the key) to authenticate PUD messages exchanged via this association in the future. The Fortezza API 56 is called to generate and wrap another sealer key to be shared with the next SNIU in the association path. The new key and this SNIU's certificate are appended to the message. The Fortezza API 56 aligns the message. The Host/Network Interface 46 transmits the message on its way back to the originating SNIU.

The originating SNIU re-assembles the Association Grant Message via the physical, data link 70, and network layers 68 as previously described. The signature is validated and audited if necessary. If valid, the Association Manager 56 uses the Fortezza API to unwrap the sealer key(s). If two keys are in the received message, the bottom key is a release key to be shared with the first intermediate SNIU; and the top key is an association key to be shared with the peer SNIU (which granted the association). If there is only one key, it is the association key which is shared with the peer SNIU; and the association path does not contain any intermediate SNIUs. Once the keys are stored and the Association Table 76 is updated, the association is established and the Session Manager 48 is called to transmit the original user datagram which was stored in the waiting Queue prior to issuing the Association Request Message.

The Session Manager 48 enforces the security policy, determines whether IP datagrams received from host computers can be transmitted via the network to their destination host, encapsulated these user datagrams in PUDs using the sealer keys for the appropriate association. The security policy is enforced by comparing the security levels of the host and destination. If the security levels are the same, the Session Manager checks the Association Table and identified the appropriate peer SNIU and sealer key(s). The user datagram is encrypted by the Fortezza API 56 using the association key. If the association contains any intermediate SNIUs, the Fortezza API 56 calculates a message authorization code using the release key. The Session Manager 48 creates a PUD addressed from this SNIU to the peer SNIU, encloses the encrypted user datagram, appends the message authorization code (if any), and passes the new datagram to the Host/Network Interface module 46 on the network-side of the SNIU. The datagram is broken into packets and transmitted as previously described.

If an intermediate SNIU receives the PUD, the data is passed through the data link layer software 70 to the network

layer where the re-assembled datagram is passed to the Session Manager 48. The source IP address is to identify the release key which is shared with the previous SNIU. The Fortezza API 56 uses the release key to verify the message authorization code. If not valid, the Session Manager 48 deletes the datagram and calls the Audit Manager 52 to generate an Audit Event Message. If the code is valid, it removes the code from the datagram, and uses the destination IP address to identify the release key shared with the next SNIU. The Fortezza API 56 generates a new message authorization code. The Session Manager 48 appends the new code and passes the datagram to the opposite port's Host Network Interface module.

When the peer SNIU (i.e., the destination IP address) received the PUD and it has been reassembled into a datagram, the Message Parser 50B passes the datagram to the Session Manager 48. The source IP address is used to identify the corresponding association key. The Fortezza API 56 decrypts the original user datagram. The Session Manager checks the message authorization code and the security levels of the source and destination hosts. If the code is valid (i.e., the message was not modified during transmission over the network) and the security levels match, the decrypted datagram is passed to the Host/Network Interface 46 to be released to the destination host. If either is not correct, the Audit Manager 52 is called.

The following is a discussion of the protocol which is applicable to both the hardware and software embodiments of the SNIU:

30 Address Resolution Messages

Address Resolution Protocol (ARP) allows a SNIU to locate a host or another SNIU, given its IP address. As previously discussed, this function is performed by the host/network interface in the hardware SNIUs and by the computer host itself for the software SNIUs. The SNIU broadcasts an ARP Request message which contains its hardware and IP addresses and the IP address of the target host. The target host (or other SNIU) returns to the requesting host an ARP Response message which contains the hardware address of the target host (or other SNIU).

Reverse Address Resolution Protocol (RARP) allows a SNIU which only knows its hardware address to obtain an IP address from the network. The host broadcasts a RARP Response which contains its hardware address, and a server on the network returns a RARP Response containing an IP address assigned to the requester's hardware address. All ARP and RARP messages have the same format and are contained within the frame data area of a single Ethernet frame (they are not IP datagrams). The format of the ARP and RARP messages is shown in the Table of FIG. 4.

Referring to FIG. 4, the HARDWARE TYPE is set to 0001 hex to indicate Ethernet. The PROTOCOL TYPE is set to 0800 hex to indicate IP addresses. The HLEN (hardware address length) is set to 06 hex bytes, while the PLEN (protocol address length) is set to 04 hex bytes. The OPERATION is set 0001 hex for an ARP request message, 0002 hex for ARP response message, 0003 hex for an RARP request message or 0004 hex for RARP response message. The SENDER'S HA contains the sender's 48 bit Ethernet hardware address, while the SENDER'S IP contains the sender's 32 bit IP address. The TARGET'S HA contains the target's 48 bit Ethernet hardware address, while the TARGET'S IP contains the sender's 32 bit IP address.

When a SNIU broadcasts a request message, it fills in all of the data and the target's hardware address field is set to 000000 hex for an ARP message. If the message is a RARP, then the sender's and target's IP address fields are set to

0000 hex. When the target machine responds, it fills in the missing address and changes the operation field to indicate a response message. During an ARP, the target machine swaps the sender's and target's addresses so that the sender's address fields contains its addresses and the target's address fields contains the original requesting host's addresses. During a RARP, the server stores its addresses in the sender's address fields and returns the response to the original sender's hardware address.

When a SNIU receives a message, it performs the following processes:

ARP Request—If an ARP Request message is received on a SNIU's port A, the untrusted software in port A's memory segment determines if the sender's IP address is in port A's ARP cache. If not, it creates a new entry in the ARP cache and inserts the sender's hardware and IP addresses. Otherwise, the sender's hardware address is copied into the entry (overwriting any previous address) and packets (if any) waiting to be sent to the sender's IP address are transmitted. If the target's IP address is in port A's address list (i.e., a List of IP addresses which are accessed from port B), the untrusted software returns an ARP Response message swapping the SENDER's and TARGET's addresses and inserting port A's Ethernet hardware address into the SENDER's HA field. In either case, the untrusted software passes the ARP Request Trusted Computing Base (TCB).

The TCB checks port B's address list for the SENDER'S IP. If the SENDER'S IP is not in port B's address list, the TCB determines whether the SENDER'S IP is releasable to port B. If releasable, the TCB inserts SENDER'S IP into port B's address list. Secondly, the TCB determines whether a proxy ARP Request should be broadcast from port B. If an ARP Response message was not returned by port A and the target's IP address is not in port A's ARP cache, then the sender's IP is releasable to port B. Thus, causing the TCB to create a proxy ARP Request message. The TCB inserts port B's hardware and IP addresses in the SENDER'S address fields, copies the target's IP address from the original ARP Request into the TARGET'S IP field and then signals port B's untrusted software to broadcast the message.

Each time the TCB releases a proxy ARP Request, it creates an Anticipated Message in the form of a proxy ARP Response message. This proxy ARP Response message contains the original sender's addresses in the TARGETS fields, the target's IP address in the SENDER'S IP field and port A's hardware address in the SENDER'S HA field. This message is saved in the Anticipated Message list for port A and will be released to port A's untrusted software for transmission, if the anticipated ARP Response message is received on port B. Note that whether this message is released may involve the TCB modulating ARP Requests from a high network to a low network in order not to exceed the 100 bits per second covert channel bandwidth requirement.

ARP Response—If an ARP Response message is received on a SNIU's port A, the untrusted software in port A's memory segment determines if the sender's IP address is in port A's ARP cache. If not, it creates a new entry in the ARP cache and inserts the sender's hardware and IP addresses. Otherwise, the sender's hardware address is copied into the entry (overwriting any previous address) and packets (if any) waiting to be sent to the sender's IP address are transmitted. Finally, the untrusted software passes the ARP Response to the TCB.

The TCB checks port B's address list for the SENDER'S IP. If the SENDER'S IP is not in port B's address list, the TCB determines whether the SENDER'S IP is releasable to

port B. If releasable, the TCB inserts it into port B's address list. Secondly, the TCB checks the Anticipated Message list for port B and determines whether the ARP Response was due to a proxy ARP Request made for a request originally received on port B. If the SENDER'S IP matches an entry in the Anticipated Message List and the message is releasable to port B, the TCB signals port B's untrusted software to create a proxy ARP Response message identical to the Anticipated Message and then removes the message from the Anticipated Message list for port B.

RARP Request—If a RARP Request message is received on a SNIU's port A, the untrusted software in port A's memory segment checks a flag to determine if the SNIU was initialized to act as a RARP server for the network attached to port A. If not, the received message is ignored. Otherwise, the untrusted software passes the RARP Request to the TCB.

The TCB determines whether the RARP Request can be released to port B. If releasable, it creates a proxy RARP Request message copying the TARGET'S HA from the received message and inserting port B's addresses in the SENDER'S HA and IP fields. Then the TCB passes the proxy RARP Request message to port B's untrusted software for broadcast and creates an Anticipated message in the form of a proxy RARP Response message. The TCB copies the original TARGETS HA, inserts port A's hardware address in the SENDER'S HA and saves it in the Anticipated Message list for port A.

RARP Response—If a RARP Response message is received on a SNIU's port A, the untrusted software in port A's memory segment determines if the sender's IP address is in port A's ARP cache. If not, it creates a new entry in the ARP cache and inserts the sender's hardware and IP addresses. Otherwise, the sender's hardware address is copied into the entry (overwriting any previous address) and packets (if any) waiting to be sent to the sender's IP address are transmitted. Finally, the untrusted software inserts the TARGET'S IP into port A's address list and passes the RARP Response to the TCB.

The TCB checks port B's address List for the SENDER'S IP. If the SENDER'S IP is not in port B's address list, the TCB determines whether the SENDER'S IP is releasable to port B. If releasable, the TCB inserts it into port B's address list. Secondly, the TCB determines whether the TARGETS IP is releasable to port B. If releasable, the TCB creates a new entry in port B's ARP cache and inserts the TARGETS HA and IP. The TCB uses the TARGETS HA to find the appropriate proxy RARP Response message in port B's Anticipated Message List and copies the TARGETS IP and SENDER'S IP into the Anticipated message. Then the TCB signals port B's untrusted software to create a proxy RARP Response message identical to the Anticipated Message, and removes the message from the Anticipated Message list for port B.

Association Establishment Messages

SNIUs establish associations in order to authenticate each other, exchange security parameters, and establish trusted sessions for communication. The SNIUs utilize a standard ICMP Echo Request message to request an association and an ICMP Echo Reply message to grant an association.

When a host behind a SNIU attempts to communicate with someone else over the network, the SNIU stores the datagram from the host in a waiting queue and transmits an ICMP Echo Request to the intended destination. This message is used to identify other SNIU units in the communications path and to carry the originating SNIU's security parameters. The SNIU inserts the originating host's security level, appends its certificate and then signs the message.

Each SNIU unit which receives this message authenticates the message, saves a copy of the previous SNIU's certificate, appends its certificate, and signs the message before sending it to the destination.

The destination host returns an ICMP Echo Reply to the originating SNIU. The first SNIU to receive this message is the terminating SNIU (i.e., closest to the destination) in the potential association's communications path. This SNIU determines if the association should be permitted (i.e. would not violate the security policy). If permitted, the SNIU grants the association, generates an encryption key for the association, and encrypts the key using the originating SNIU's public key (from its certificate). If the Echo Request message contained an intermediate SNIU's certificate, the SNIU also generates a release key and encrypts it using the intermediate SNIU's public key. In either case, the SNIU removes its and the originating SNIU's security parameters and signatures from the ICMP Echo Reply, stores the encrypted key(s), inserts the destination host's security level, appends its certificate, signs the message and sends it onto the originating SNIU.

Each intermediate SNIU (if any exist) which receives the Echo Reply message authenticates the previous SNIU's signature, extracts the release key, generates a new release key for the next SNIU, encrypts the key using the public key (from the certificate saved from the Echo Request message) of the next SNIU, removes the previous intermediate SNIU's certificate and signature, appends its own certificate and signature, and sends the message on the return path. When the originating SNIU receives the ICMP Echo Reply, it authenticates the message and extracts the key(s).

Once the association is granted, the originating SNIU fetches the originating host's datagram from the waiting queue and prepares to send it to the terminating SNIU in the newly established association. The SNIU uses the association key to encrypt the datagram for privacy. The SNIU further stores the encrypted datagram and the encryption residue into a new datagram from the originating SNIU to the terminating SNIU. If the association contains intermediate SNIUs, the originating SNIU uses the release key to calculate a second encryption residue and appends it to the datagram. Finally, the SNIU transmits the protected user datagram to the peer SNIU in the association.

When the protected user datagram is received by an intermediate SNIU (if any in the path), the intermediate SNIU fetches the release key corresponding to the previous SNIU and uses the release key to validate the datagram. If valid, the SNIU removes the release key residue from the datagram and checks to determine whether there are more intermediate SNIUs in the path before reaching the terminating SNIU. If another intermediate SNIU exists, the release key corresponding to the next intermediate SNIU is used to calculate a new release residue which is appended to the datagram. In either case, the datagram is sent on its way out the opposite port from which it was received.

When the terminating SNIU receives the protected user datagram, it uses the association key corresponding to the originating SNIU to decrypt and validate the datagram. If the source and destination hosts are at the same security level (i.e., a write-equal situation), the decrypted datagram is sent out the opposite port to the destination host. If the source host has a lower security level than the destination (i.e., a write-up situation), the SNIU predicts the response from the destination and saves it before sending the decrypted datagram to the destination host.

If the source host has a higher security level than the destination (i.e., a write-down situation), the received data-

gram (i.e., a response to a previous datagram from the lower level host) was predicted by the SNIU which sent the protected datagram. Therefore, this SNIU is assured that the classification of the received datagram is dominated by the lower Level destination host, so that the datagram is released to the destination. If a SNIU receives a user datagram from a native host which would be a write-down to the destination host and no predicted datagram is found, the received datagram is erased and the attempted write-down is audited.

10 Message Processing Tables

There are three tables which are used to process in-coming and out-going messages including the Association Table, the Symmetric Key Table(Sym_Key) and the Certificate Table. Each SNIU has to Association tables, one for each port. Each entry contains data corresponding to a particular source or destination IP address. The Sym_Key table contains data corresponding to a particular message encryption key (MEK) which could be used as a release key or an association key. The Certificate table contains recently received certificates from other SNIU's.

Each table consists of a linked list of tokens in which the data for an entry in the table is stored in a token. The tokens for each table have a unique data structure and are linked together in 'free' lists during initialization. When a new entry is made in one of the tables, a token is removed from the free list for that table's tokens, the data for the new entry is inserted in the appropriate fields of the token and the token is linked at the top of the table. When an entry is removed from a table, the 'previous' and 'next' tokens are linked, the data fields in the token are cleared and the token is linked at the bottom of the appropriate free list. Whenever the data in an entry is used, the token is removed from the table and relinked at the top of the table. In this way the oldest (i.e., least used) entry is at the bottom of the table.

If a new entry is needed and the free list is empty, the bottom token is removed from the table, the data fields are cleared, the new entry's data is inserted and the token is linked at the top of the table. In addition, when a SNIU removes the bottom (oldest unused) token in the Sym_Key Table, it also removes every token in the Association Table which pointed to the removed key. A SNIU does not send a Close Association Message when a certificate, key or Association Table entry is removed because many valid entries using the same association may still exist. The data structure for a token of the Association Table is shown in the Table of FIG. 5.

Referring to FIG. 5, NEXT is a pointer to the next token in the table or list. PREVIOUS is a pointer to the previous token in the table or list. IP ADDRESS is the IP address of the source/destination, while PEER SNIU IP ADDRESS is the address of the other terminating SNIU for the association. ASSOCIATION KEY POINTER points to the association MEK in the Sym_Key table. RELEASE KEY POINTER points to the release MEK in the Sym_Key table. The ASSOC-TYPE is set to 0001 hex for 'pending', 0002 hex for 'host'(i.e., the entry is for a host destination), 0003 hex for 'sniu'(i.e., the entry is for a SNIU destination), 0004 hex for 'native host'(i.e., no peer SNIU) or 0005 hex for 'audit catcher'. The RELKEY-TYPE is set to 0001 hex for 'in'(i.e., use to validate release key residue), 0002 hex for 'out'(i.e., use to add release key residue) or 0003 hex for 'both'. SECURITY-LEVEL indicates the security level of the source/destination, while SPARE is an unused byte to keep addressing on a 32-bit boundary.

Referring to FIG. 6, there is shown the data structure of a token for the Sym_Key Table according to the present invention. NEXT is a pointer to the next token in the table

or list, while PREVIOUS is a pointer to the previous token in the table or list. DISTINGUISHED NAME is the 128 byte name in certificate from the other SNIU using this key. MEK is the 12 byte wrapped key (association or release) shared with the another SNIU. IV is the 24 byte initialization vector associated with the MEK. CERTIFICATE POINTER points to the other SNIU's certificate in the Certificate table. INDEX is a Fortezza card key register index which indicates if and where the key is loaded (1-9 are valid key register indexes, while 0 indicate that the key is not loaded on the Fortezza). SPARE is an unused byte to keep addressing on a 32-bit boundary.

Message Flag

Any message (IP datagram) which is generated or modified by a SNIU unit contains a Message Flag in the last four bytes of the datagram. The first byte is the message type field, the second byte is the message format field and the third and fourth bytes are the Flag. Note that all message types are signed except for a Protected User Datagram (PUD) which uses MEK residues for integrity and authentication.

Waiting Queue and Schedule Table

The Waiting Queue is used to store IP datagrams for potential future processing based on an anticipated event. For every entry made in the Waiting Queue, a corresponding entry is made in the Schedule Table. The Schedule Table is used to automatically process entries in the Waiting Queue if they have not been processed within some predetermined amount of time (i.e. the anticipated event does not occur). The Schedule Table entry contains a time-out field (which is set to the current time plus some reasonable delta representing the maximum waiting period) and a function pointer (which indicates which subroutine should be called if time expires before the Waiting Queue entry is processed). The Schedule Table is checked in the main executive loop of the TCB, expired entries are removed and the corresponding datagrams in the Waiting Queue are processed by the designated subroutine.

For example, when a SNIU receives a user datagram from a native host which is destined for another host for which there is no existing association, the SNIU stores the user datagram in the Waiting Queue and transmits an Association Request message. When the Association Grant message is received, the user datagram is removed from the Waiting Queue, the corresponding Schedule Table entry is deleted, the user datagram is encrypted and sent to the peer SNIU of the association. If an Association Grant message is never received, the Schedule Table entry expires which calls a subroutine to delete the user datagram from the Waiting Queue.

Another example is when the SNIU sends an Audit Event message to an Audit Catcher. The transmitted datagram is stored in the Waiting Queue. When the Receipt message is received from the Audit Catcher, the original Audit Event datagram is removed from the Waiting Queue and the corresponding Schedule Table entry is deleted. If the Schedule Table entry expires, the designated subroutine is called which re-transmits Audit Event message stored in the Waiting Queue and a new entry is made in the Schedule Table.

Generating and Exchanging MEKs

Message Encryption Keys (MEKs) are generated during the association establishment process (previously described) and are exchanged via the Association Grant Message. When a SNIU generates an MEK, it simultaneously generates an initialization vector (IV).

When a SNIU exchanges an MEK with another SNIU, it generates a random number, RA, which is required to

encrypt (i.e., wrap) the MEK. The key exchange algorithm is designed so that only the sending and receiving SNIUs can decrypt the MEK and use it. The sender wraps the MEK for transmission using the destination's public Key RA, RB (which is always set =1) and the sender's private key. IVs which were generated with release keys are transmitted in the clear with the wrapped MEK in the Association Grant Message, while IVs which were generated with association keys are ignored. The recipient unwraps the key using its private key RA, RB, and the sending SNIU's public key. Once unwrapped, the safe exchange is complete.

Each SNIU re-wraps the MEK using its storage key (Ks), stores the MEK and the IV (if the MEK is a release key) in the Sym_Key Table, stores the pointer to the MEK in the Association Table and stores the DN (of the other SNIU sharing this MEK) in the Sym_Key Table entry.

Using MEKs and IVs

Message Encryption Keys (MEKs) are used as association and release keys to provide confidentiality, integrity and authentication of user datagrams during an association between two SNIUs. IVs are used to initialize the feedback loop in the Skipjack encryption algorithm for most modes of operation. Encrypting identical data using the same MEK, but different IVs, will produce different cipher text. In fact, the Fortezza card requires the user to generate a new IV for each encryption event in order to assure that each message looks different when encrypted.

When a SNIU encrypts a user datagram it first generates a new IV for the association key, encrypts the datagram, appends the encryption residue for integrity and authentication purposes, and appends the new IV. If the association involves intermediate SNIUs, the SNIU performs a decrypt operation on the newly encrypted datagram, residue and IV. The decrypt operation uses the release key and release Key IV. The release key IV is never changed since the encrypted data is always guaranteed to be unique even if the original datagram is not. The release key residue is appended to the protected user datagram. The completed protected user datagram is then transmitted.

Received Message Processing

When a SNIU receives an IP datagram, it checks the destination address in the header and determines if it is the intended recipient. Then, the SNIU checks the last four bytes of the IP datagram for the Message flag and determines the type and format of the received message.

Destination SNIU Message Processing

When a SNIU receives an IP datagram which is addressed to it, the message should be one of the following messages: an Audit Event, Audit Catcher list, Audit Mask, Association Close, Association Request, Association Grant, Association Denial, Association Unknown, Protected User Datagram, Receipt and Certificate Revocation List. If it is not, the SNIU audits the event. The only exceptions are ICMP datagrams which are processed by the receiving port's untrusted software and not passed to the trusted computing base.

Audit Event—If the SNIU is not configured to be an Audit Catcher, it will audit the event sending the source IP address of the received message to its primary Audit catcher.

If the SNIU is configured to be an Audit Catcher, it verifies the signature on the message, increments its received audit event sequence number, generates a time stamp, and prints the sequence number, time stamp, source IP address, and ASCII character string from the message. Once the event has been recorded, the Audit catcher SNIU generates a Receipt Message (copies the audit event counter from the received message and inserts it in the message number field), sends it, and checks the receiving port's

Association Table for an entry for the source of the received message. If an entry does not exist (i.e., this was the first communication from the source SNIU), the Audit Catcher makes an entry, marks the association type as 'sniu', and sends the SNIU the current audit mask.

Audit Catcher List—The SNIU verifies the signature on the message, stores the new list of Audit catchers in the Configuration table, generates a receipt message, and audits the event.

Audit Mask—the SNIU verifies the signature on the message, stores the new audit mask in the Configuration Table, generates a Receipt Message, and audits the event (in case someone else other than the Audit Catcher is distributing new audit masks). In addition, if the receiving SNIU is an Audit Catcher, it distributes the new audit mask to every destination in the Association Table with an association type of 'sniu'.

Association Close—When a SNIU receives a valid protected User Datagram, but cannot find the destination's Association Table entry, it sends an Association close message back to the originating SNIU and audits the event. The originating SNIU verifies the signature on the received Association Close Message, extracts the original destination host's IP, removes the host's entry from its Association Table and audits the event. It does not remove the peer SNIU's entry nor entries from the Sym_Key table as they might be supporting other associations.

Association Request—This message can only be received by a SNIU which originally transmitted it as an ICMP echo request to some unknown destination which converted it to an Echo reply and returned it to the originator without encountering another SNIU. Therefore, the SNIU uses the source IP address to find the destination's entry in the Association Table, changes the association type from 'pending' to 'native host', sets the security level to that port's security level, finds the original host's user datagram in the Waiting Queue, removes the corresponding entry from the Schedule table, and compares the source and destination security levels to determine if the user program can be sent to the destination. If the comparison indicates a write-up situation, the SNIU generates and saves an anticipated message and releases the original datagram to the destination port. If a write down situation, the SNIU determines if the data gram was predicted and sends the anticipated message or audits as previously described. If a write equal, the datagram is released to the destination port. This procedure is repeated for each entry in the Waiting Queue which is intended for the same destination. **Association Grant**—The SNIU verifies the signature in the datagram and updates the receiving port's Association Table entries for the host destination and peer SNIU. The SNIU finds the entry for the destination host, changes the association type from 'pending' to 'host', copies the peer SNIU's IP, extracts and unwraps the association MEK (and release MEK if needed), stores the re-wrapped key(s) in the Sym_Key table, marks the release key type as 'out' (if a release key exists), copies the destination host's security level, and determines if an entry exists for the peer SNIU. If not, the SNIU creates a new entry for the peer SNIU, copies the association and release key pointers and release key type from the destination host's entry, and marks the association type as 'sniu'.

Once receiving the port's Association Table has been updated, the SNIU finds the original hosts user datagram in the waiting Queue, removes the corresponding entry from the Schedule table, and compares the source and destination security levels to determine if the user datagram can be sent to the destination. If the source's security level is dominated

by (i.e., less than or equal to) the destination's security level, the SNIU creates a Protected user Datagram (PUD). The SNIU sets the destination to the peer SNIU's IP, sets the protocol type to indicate a SNIU Message, uses the association key to encrypt the entire received datagram, inserts the ciphertext and IV, appends the association residue, generates and inserts a release residue (if the destination host's Association Table entry contains a pointer to a release key), appends the appropriate SNIU Message Flag, and sends the datagram. If the source host is not dominated by the destination (i.e., potential write down, the attempted write down is audited. This procedure is repeated for each entry in the Waiting Queue which is intended for the same destination.

Association Unknown—A SNIU sends an Association Unknown message (and generates audit notices) when a protected user datagram is received and a corresponding Association Table entry does not exist. The message is sent back to the source SNIU and contains the destination SNIU's IP address. When a SNIU receives an Association Unknown Message, it deletes every entry in the Association Table in which the peer SNIU address matches the returned destination SNIU IP. Subsequent user datagrams from the same host sent to the same destination will initiate an Association Request to re-establish the association. Any SNIU involved in establishing an association for which it already has keys (association and/or release keys) will suggest the same key as originally used.

Protected User Datagram—the SNIU uses the source IP to find the appropriate entry in the receiving port's Association Table and retrieve the association key to decrypt and validate the received datagram. If the decryption residue does not match, the even is audited. Otherwise, the SNIU uses the destination host's IP to find the appropriate entry in the opposite port's Association Table, retrieves the destination host's security level, and compares it to the security level in the received datagram. If a write-up situation, the SNIU generates an anticipated message. However, regardless of the relative security levels, the decrypted and validated user datagram is sent to the destination host.

If a terminating SNIU receives a PUD and validates the residue but cannot deliver the user datagram because it cannot find the destination host in the Association Table, then the SNIU returns an Association close message to the originating SNIU (containing the destination host's IP) and audits the event.

Receipt—A Receipt message is sent by an Audit catcher to a SNIU for Audit Catcher Request and Audit Event messages. The SNIU uses the message number in the received datagram to locate the saved copy of the original message in the Waiting Queue and remove it and the corresponding Schedule Table entry. If the original message was an Audit Catcher Request Message, the SNIU locates the Association Table entry for the Audit catcher and changes the association type from 'pending' to 'audit catcher'. If time expires in the Schedule Table entry before the Receipt Message is received, the SNIU will retransmit the original message. If no receipt is received after TBD attempts, the SNIU will switch to the next Audit Catcher in the list. If all Audit Catchers are attempted without success, the SNIU will check a configuration parameter to determine whether to continue without audit or halt.

SNIUs issue Receipt messages to Audit catchers for Audit Catcher List, Audit Mask, and Certificate Revocation List messages. When an Audit Catcher receives a receipt, it uses the returned message number to remove the copy of the message from the Waiting Queue and the corresponding Schedule table entry.

Certificate Revocation List—if a Certificate revocation List (CRL) is received, the SNIU returns a receipt to the source and checks the Sym_Key Table for any keys which were received from (or sent to) another SNIU with a revoked certificate, the SNIU deletes the certificate from the Certificate Table (if it is still there), deletes the Sym_Key Table entry, and deletes every entry in the Association Table which pointed to the key. Note that deleting a table entry means to unlink the token from the table, clear the token's memory, and re-link the token in the token's free list.

Non-Destination SNIU Message Processing

When a SNIU receives an IP datagram which is not addressed to it, the message should be one of the following types of Dragonfly formatted messages. If it is not, the SNIU will assume the IP datagram is from a native host.

Audit Event—The SNIU verifies the signature on the message and releases the message out the opposite port.
Audit Catcher List—The SNIU verifies the signature on the message and releases the message out the opposite port.

Audit Mask—The SNIU verifies the signature on the message and releases the message out the opposite port.

Association Close—The SNIU verifies the signature on the message and releases the message out the opposite port.

Association Request—When a SNIU receives an Association Request it first checks the IP header to determine if the datagram is an ICMP Echo Request or an ICMP Echo Reply.

If it is an ICMP Echo Request, the SNIU validates the signature at the bottom of the message and checks the receiving port's Association Table for an entry with the originating SNIU's IP address. If the receiving SNIU cannot find an entry, it creates one, marks the association type as 'pending', stores the previous SNIU's certificate in the Certificate Table (if it wasn't already there), updates the Sym_Key Table entry for the Distinguished Name (DN), and stores the pointer to the Sym_Key Table entry in the release key pointer field in the Association Table entry. If the previous SNIU was an intermediate SNIU (i.e., the Message Format field of the SNIU Message Flag is 'Signed Type 2'), this SNIU marks the release key type field as 'out' and removes the previous SNIU's certificate and signature. In either case, this SNIU appends its certificate and signature and sends the message out the other port. It does not make any entry in the out-going port's Association Table.

If it is an ICMP Echo Reply, this SNIU is the terminating SNIU which must generate the Association Grant Message. Before the SNIU can validate the received message, it must reconstruct the message to match the form it was in when the SNIU signed it before the destination host converted the ICMP Echo Request into an ICMP Echo Reply. Therefore, the SNIU exchanges the source and destination IP addresses in the datagram header and changes the type field in the ICMP header from a request (8) to a reply (0). Then the SNIU validates the signature at the bottom of the newly reconstructed message using its own public key. If the signature cannot be validated, the event is audited.

If the ICMP Echo Reply is valid, the SNIU creates or updates three Association Table entries. First, it creates an entry (if it doesn't already exist) in the receiving port's Association Table for the original destination host (using the destination IP from the modified datagram header), marks the association type as 'native host' and stores the receiving port's security level in the security level field. Second, it updates the entry in the opposite port's Association Table for the peer SNIU (using the source IP from the modified

datagram header). If the release key type is marked 'out' or 'both', then the association path contains at least one intermediate SNIU; therefore, the SNIU extracts the peer SNIU's certificate from the datagram, stores it in the Certificate Table, stores the pointer to the certificate and the DN in a Sym_Key Table entry, and stores the pointer to the Sym_Key Table entry in the association key pointer field of the Association Table entry. If there aren't any intermediate SNIUs, the pointer in the release key pointer field is copied to the association key pointer field; and the release key pointer field is cleared. In either case the association type is marked as 'sniu'. The third Association Table entry is for the originating host. Its IP and security level are in the data portion of the received datagram. The security level is copied into the entry, the association type is marked as 'host', and the rest of the data is copied from the peer SNIU entry. Finally, the SNIU generates the association key (and if necessary, the release key) and stores the key(s) in the Sym_Key Table entry(s).

Once the Association Table entries are updated, an Association Grant Message is generated. The SNIU uses the peer (i.e., originating) SNIU's IP for the destination, uses the original destination host's IP for the source, and marks the protocol and type fields to indicate an ICMP Echo Reply. The SNIU inserts its IP address, its certificate, its host's security level, the association key data (wrapped key and RA), and if necessary, the release key data (the wrapped key, RA and IV). The SNIU Message Flag is inserted at the bottom marking the type as Association Grant and the format as Signed Type 1 to indicate only one certificate. The message is signed and sent.

Association Grant—The SNIU validates the signature at the bottom of the received datagram and, if not correct, audits the event. Otherwise, since it is not the destination, the SNIU is an intermediate SNIU somewhere in the path between the two peer SNIUs. The SNIU creates an entry (if one doesn't already exist) in the receiving port's Association Table for the IP of the terminating SNIU which granted the association (note that the terminating SNIU's IP is not in the header of the received datagram, rather it is in the data area), marks the association type as 'sniu', marks the release key type as 'in' (if the format is 'Signed Type 1') or 'both' (if the format is 'Signed Type 2'), extracts the release key data (i.e., the wrapped MEK, RA and IV), unwraps and stores the release key in the Sym_Key Table, stores the release key IV in the same Sym_Key Table entry, stores the pointer to the release key in the Association Table, stores the certificate in the Certificate Table, and stores the pointer to the certificate and the DN in the Sym_Key Table entry.

Next, the SNIU uses the destination IP address in the header of the received Association Grant Message to find the destination's entry in the opposite port's Association Table. If the association type is 'pending', the SNIU uses the release key pointer to fetch the saved certificate of the next SNIU, generates release key data (an MEK, RA, and IV), stores the wrapped MEK and IV in the Sym_Key Table entry, and changes the association type to 'sniu'. If the association type is 'NULL', the SNIU changes it to 'in'; otherwise, it is marked as 'both'.

Finally, the SNIU rebuilds the Association Grant Message to send on to the destination. The SNIU copies the received datagram up to and including the association key data and the certificate of the SNIU which originated the Association Grant Message, inserts its certificate and the release key data, and signs and sends the datagram.

Association Unknown—The SNIU verifies the signature on the message and releases the message out the opposite port.

Protected User Datagram—The SNIU uses the source IP address to find the appropriate entry in the receiving port's Association Table, fetches the release key, and verifies the release key residue. If the release residue is not correct the datagram is deleted and the event audited. Otherwise, the SNIU uses the destination IP address to find the appropriate entry in the opposite port's Association Table, fetches the release key, generates the new release residue, overwrites the old release residue, and sends the datagram on to the destination.

Receipt—The SNIU verifies the signature of the message and releases the message out the opposite port.

Certificate Revocation List —The SNIU verifies the signature on the message and releases the message out the opposite port.

Native Host Message—When a SNIU receives a user datagram from a native host, the SNIU creates an entry (if one doesn't already exist) in the receiving port's Association Table for the source host's IP, marks the association type as 'native host', sets the security level to the receiving port's security level, and checks the opposite port's Association Table for the destination's IP address.

If an entry does not already exist for the destination, the SNIU creates a new entry, marks the association type as 'pending', stores the received datagram in the Waiting Queue, makes a corresponding entry in the Schedule Table, creates an Association Request Message and sends it.

If an Association Table entry exists for the destination and the association type is 'pending', the SNIU stores the received datagram in the Waiting Queue, linking it to other datagrams for the same destination.

If an Association Table entry exists for the destination and the association type is 'host', the SNIU compares the source host's security level to the destination host's security level. If the source's security level is dominated by (i.e., less than or equal to) the destination, the SNIU creates a Protected User Datagram (PUD). The SNIU sets the destination to the peer SNIU's IP, sets the protocol type to indicate a SNIU Message, uses the association key to encrypt the entire received datagram, inserts the cipher text and IV, appends the association residue, generates and inserts a release residue (if the Association Table entry contains a pointer to a release Key), appends the appropriate Dragonfly Message Flag, and sends the datagram. If the source host is not dominated by the destination (i.e., a potential write down), the SNIU determines if this datagram was anticipated. If a matching datagram was predicted the anticipated datagram is transformed into a PUD (as described above) and sent. If an anticipated message is not found, the attempted write-down is audited.

If an Association Table entry exists for the destination and the association type is any other bona fide type, i.e., 'native host', 'sniu', or 'audit catcher', the SNIU compares the source and destination port's security levels to determine if the datagram can be allowed to proceed. If the comparison indicates a write-up situation, the SNIU generates and saves an anticipated message and releases the original datagram to the destination port. If a write-down situation, the SNIU determines if the datagram was predicted and sends the anticipated message or audits as previously described. If a write-equal, the datagram is released to the destination port.

It is to be understood that the embodiments described herein are merely exemplary of the principles of the invention, and that a person skilled in the art may make many variations and modifications without departing from the spirit and scope of the invention. All such variations and modifications are intended to be included within the scope of the invention as defined in the appended claims.

What is claimed is:

1. A multi-level network security system for a computer host device coupled to at least one computer network, comprising:

a secure network interface Unit (SNIU) contained within a communications stack of said computer device that operates at a user layer communications protocol, said SNIU communicates with other like SNIU devices on said network by establishing an association, thereby creating a global security perimeter for end-to-end communications and wherein said network may be individually secure or non-secure without compromising security of communications within said global security perimeter, comprising:

a host/network interface for receiving messages sent between said computer device and said network, said interface operative to convert said received messages to and from a format utilized by said network;
a message parser for determining whether said association already exists with another SNIU device;
a session manager coupled to said network interface for identifying and verifying said computer device requesting access to said network, said session manager also for transmitting said messages received from said computer device when said message parser determines said association already exists; and
an association manager coupled to said host/network interface for establishing an association with other like SNIU devices when said message parser determines said association does not exist.

2. The system of claim 1, wherein said SNIU is contained within said communications stack between a Network layer and a Data Link layer.

3. The system of claim 1, which further includes means coupled to SNIU for performing both encryption and decryption functions.

4. The system of claim 3, which further includes means for generating and writing cryptographic residues for outgoing messages and validating cryptographic residues for incoming residues.

5. The system of claim 1, wherein said session manager protects the security communications between said computer device and said network by implementing a security policy selected from a group consisting of discretionary access control, mandatory access control, labeling, denial of service detection, data type integrity, cascading control and covert channel use detection.

6. The system of claim 1, wherein said SNIU further includes means for performing a defined trusted session layer protocol (TSP), said TSP constituting said user layer communications protocol.

7. The system of claim 1, wherein said SNIUs exchange security parameters during said association.

8. The system of claim 1, wherein said SNIU further includes a scheduler coupled between said host/network and said message parser for controlling the flow of said data within said SNIU.

9. The system of claim 1, wherein said Association manager generates two messages in order to establish said association.

10. The system of claim 1, wherein said SNIU further includes an audit manager coupled to said association manager for generating audit event messages when a message is received with an invalid authorization code.

11. A method of providing a multi-level network security system for a portable computer device coupled to at least one computer network, comprising:

23

placing a secure network interface Unit (SNIU) within a communications stack of said computer device that operates at a user layer communications protocol, said SNIU communicates with other like SNIU devices on said network by establishing an association, thereby creating a global security perimeter for end-to-end communications and wherein said network may be individually secure or non-secure without compromising security of communications within said global security perimeter, said SNIU performing a plurality of security functions including:

- receiving said messages sent between said computer device and said network;
- converting said received messages to and from a format utilized by said network;
- identifying and verifying said computer device requesting access to said network at the session level;
- determining whether said association already exists with another SNIU device;
- transmitting said messages received from said computer device when said association already exists; and
- establishing an association with other like SNIU devices when said association does not exist.

12. The method of claim 11, wherein said SNIU is placed within said communications stack between a Network layer and a Data Link layer.

24

13. The method of claim 11, which further includes encrypting outgoing messages and decrypting incoming messages of said SNIU.

14. The method of claim 13, which further includes generating and writing cryptographic residues for outgoing messages and validating cryptographic residues for incoming residues.

15. The method of claim 11, wherein said SNIU protects the security communications between said computer device and said network by implementing a security policy selected from a group consisting of discretionary access control, mandatory access control, labeling, denial of service detection, data type integrity, cascading control and covert channel use detection.

16. The method of claim 11, wherein said SNIU further performs a defined trusted session layer protocol (TSP), said TSP constituting said user layer communications protocol.

17. The method of claim 11, which further includes generating an audit trail.

18. The method of claim 11, which further includes temporarily storing said received messages from said computer device when said association does not exist.

19. The method of claim 11, wherein said association is established by generating two messages.

20. The method of claim 1, wherein said SNIUs exchange security parameters during said association.

* * * * *



US006064304A

United States Patent [19]
Arrowsmith et al.

[11] **Patent Number:** 6,064,304
 [45] **Date of Patent:** *May 16, 2000

[54] **METHOD AND APPARATUS FOR POLICY-BASED ALARM NOTIFICATION IN A DISTRIBUTED NETWORK MANAGEMENT ENVIRONMENT**

[75] **Inventors:** Russell Arrowsmith, Merrimack, N.H.;
 William Tracy, Chelmsford, Mass.

[73] **Assignee:** Cabletron Systems, Inc., Rochester, N.H.

[*] **Notice:** This patent is subject to a terminal disclaimer.

[21] **Appl. No.:** 09/307,833

[22] **Filed:** May 10, 1999

[51] **Int. Cl.⁷** G08B 29/00

[52] **U.S. Cl.** 340/506; 709/229; 709/224;
 714/25; 714/39; 714/46; 714/48

[58] **Field of Search** 340/506; 709/229,
 709/224; 714/25, 39, 46, 48

[56] **References Cited**
 U.S. PATENT DOCUMENTS

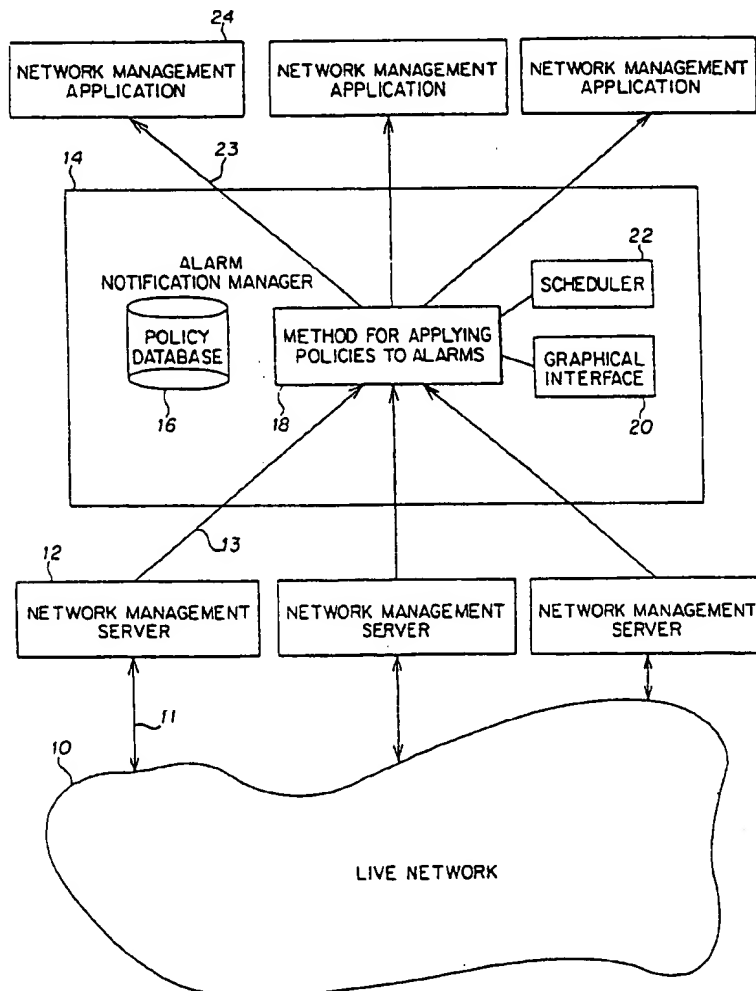
5,261,044 11/1993 Dev et al. 345/357

Primary Examiner—Daryl Pope
Attorney, Agent, or Firm—Wolf, Greenfield & Sacks, P.C.

[57] **ABSTRACT**

Apparatus and method for receiving alarms from multiple network management servers and applying a plurality of policy-based filters to the alarms. The filters may be named and stored in a database, and application of the policy-based filters may be scheduled for different times. The same policy-based filters may be applied to one or more multiple network management applications. The invention allows greater control over which alarms get reported to network management applications and provides a means to ensure consistency of reported alarms across multiple network management applications.

15 Claims, 9 Drawing Sheets



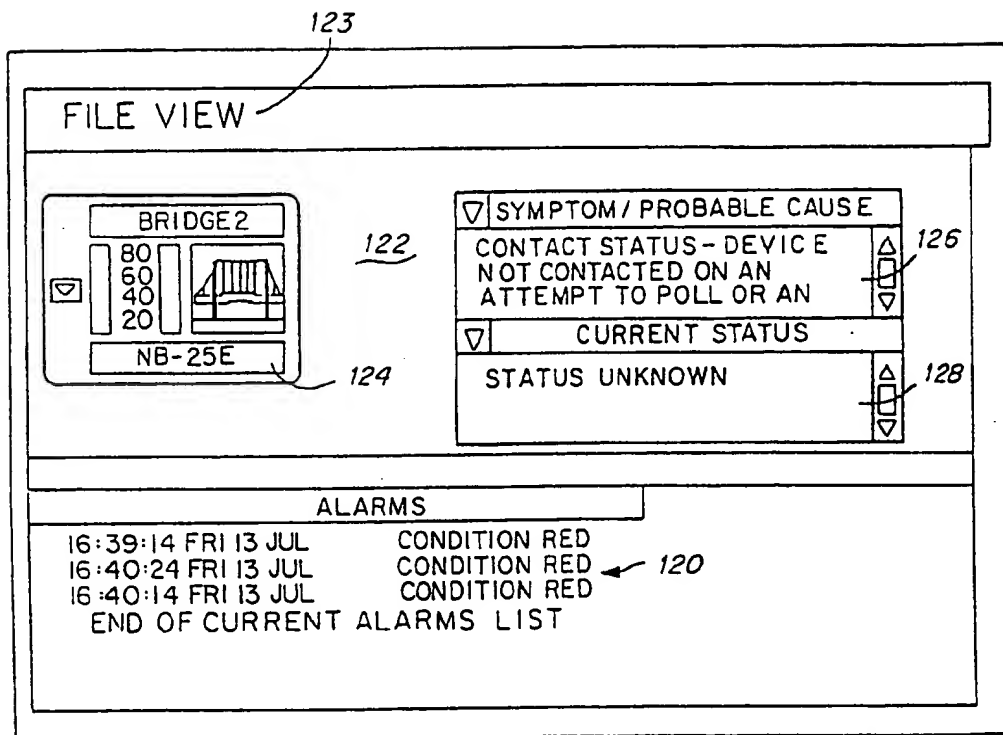


FIG. 1
(PRIOR ART)

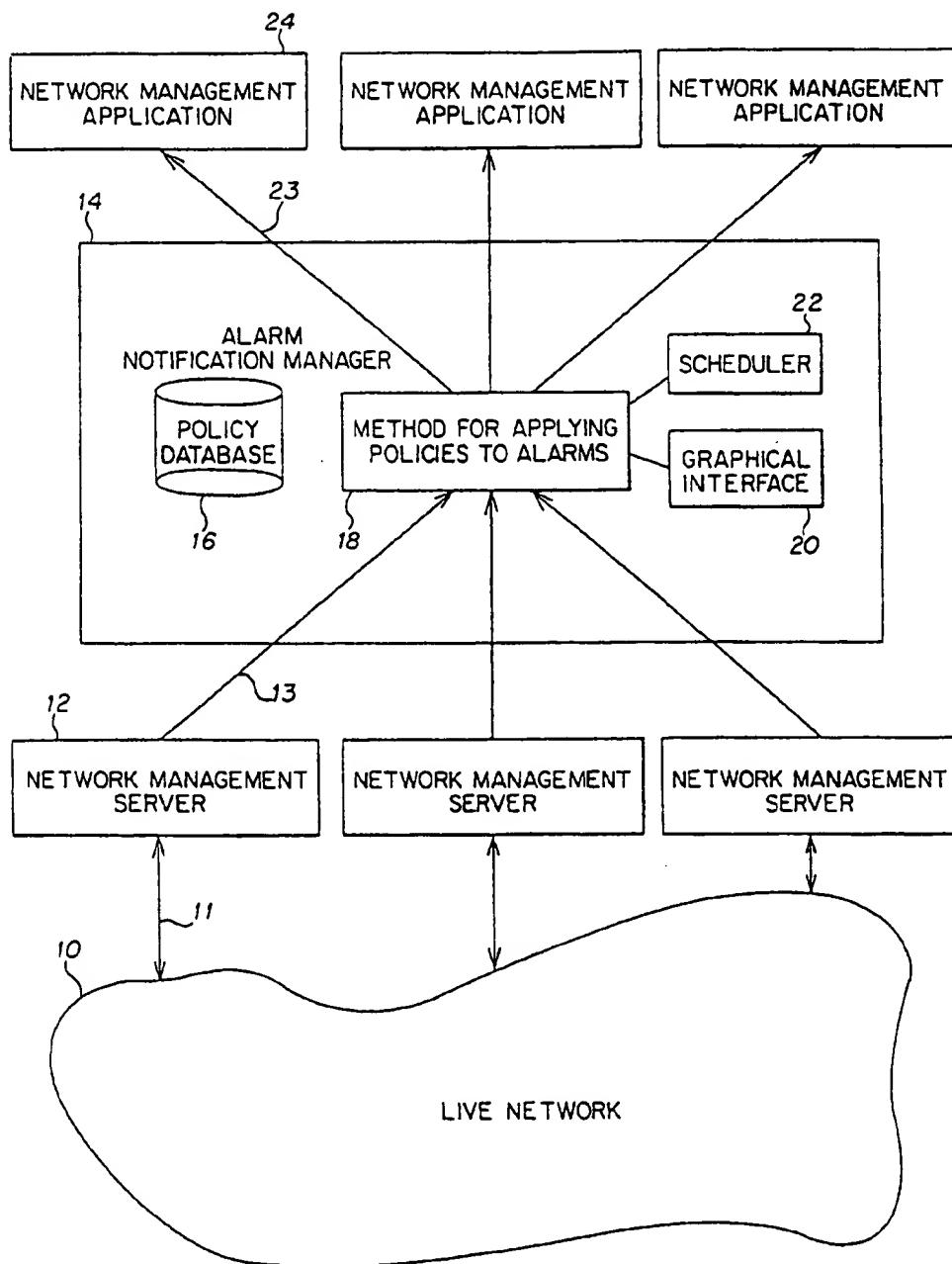
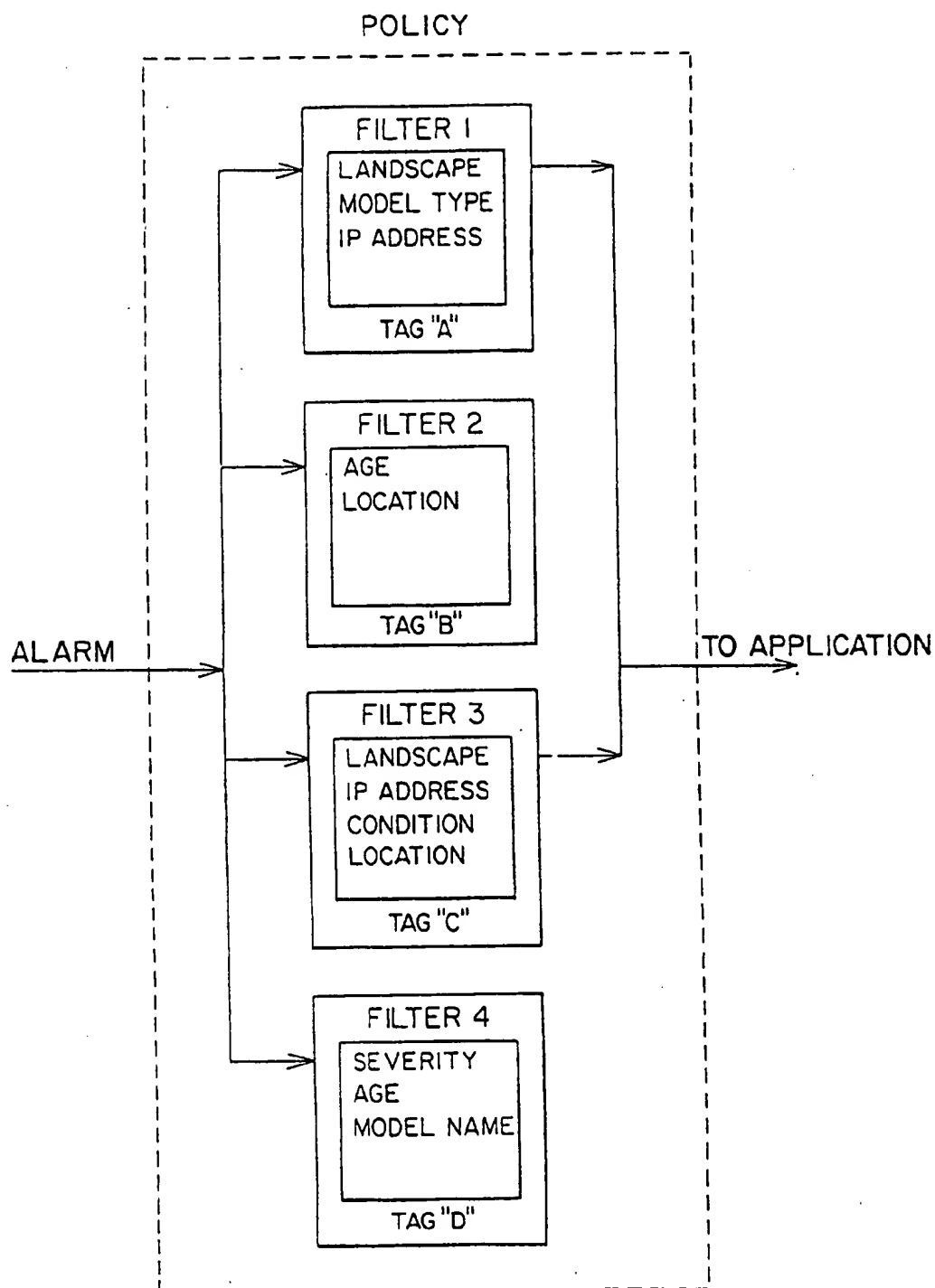


FIG. 2

**FIG. 3**

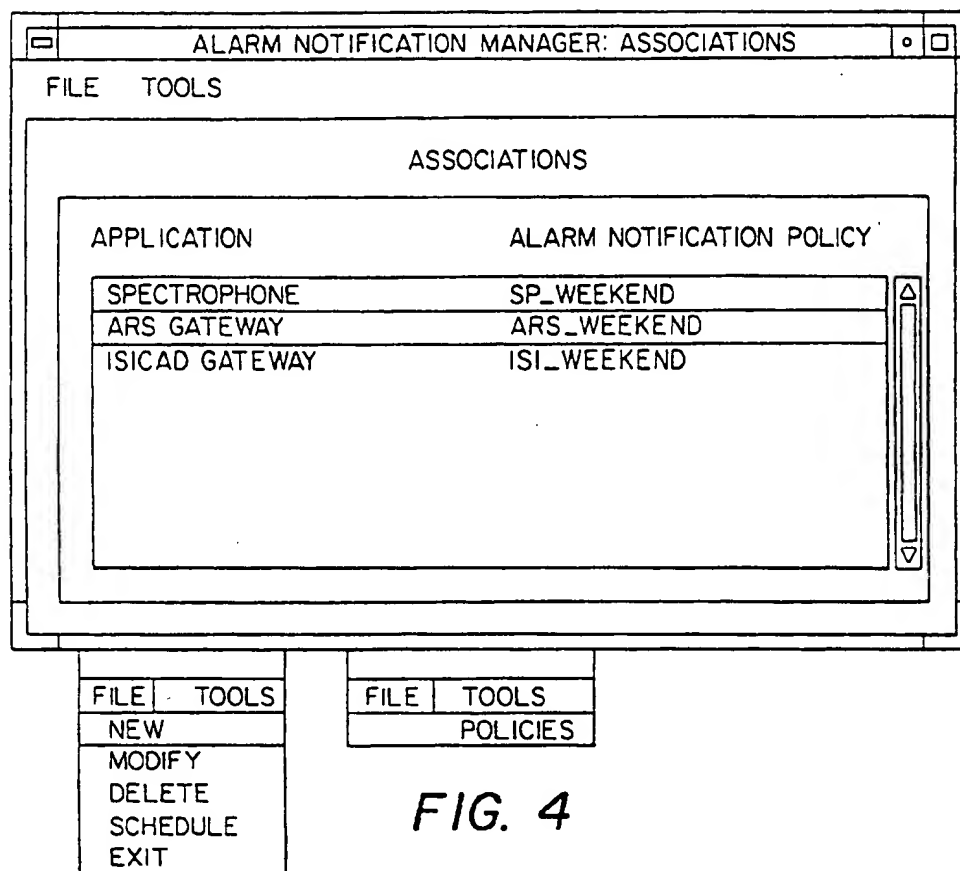


FIG. 4

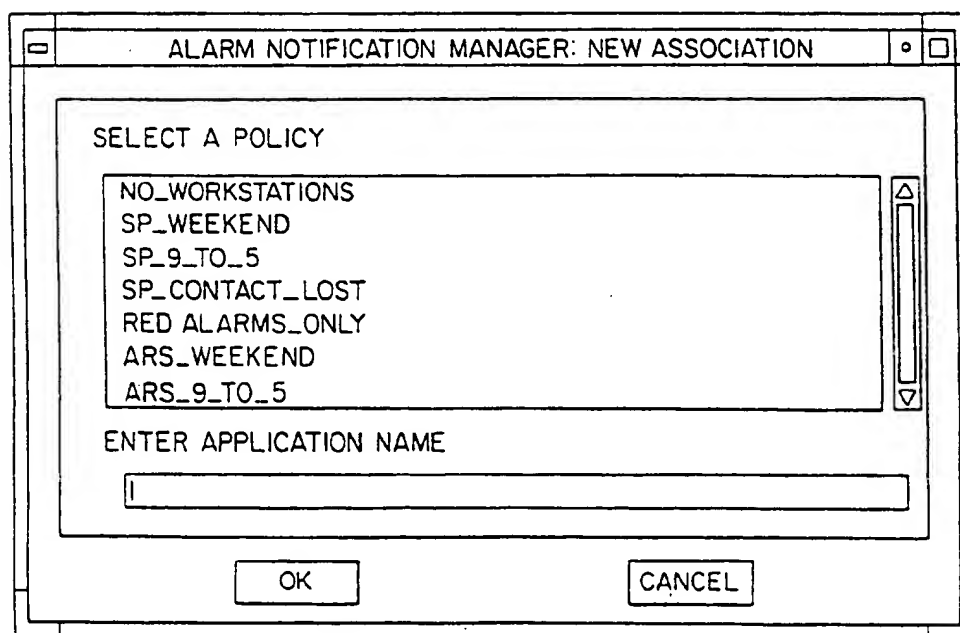


FIG. 5

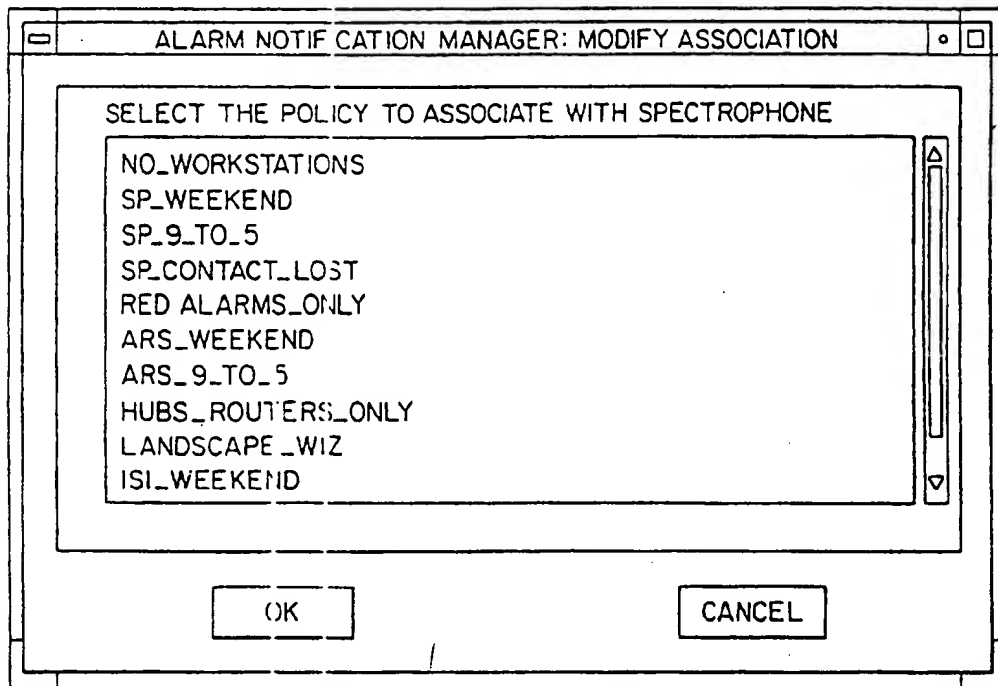


FIG. 6

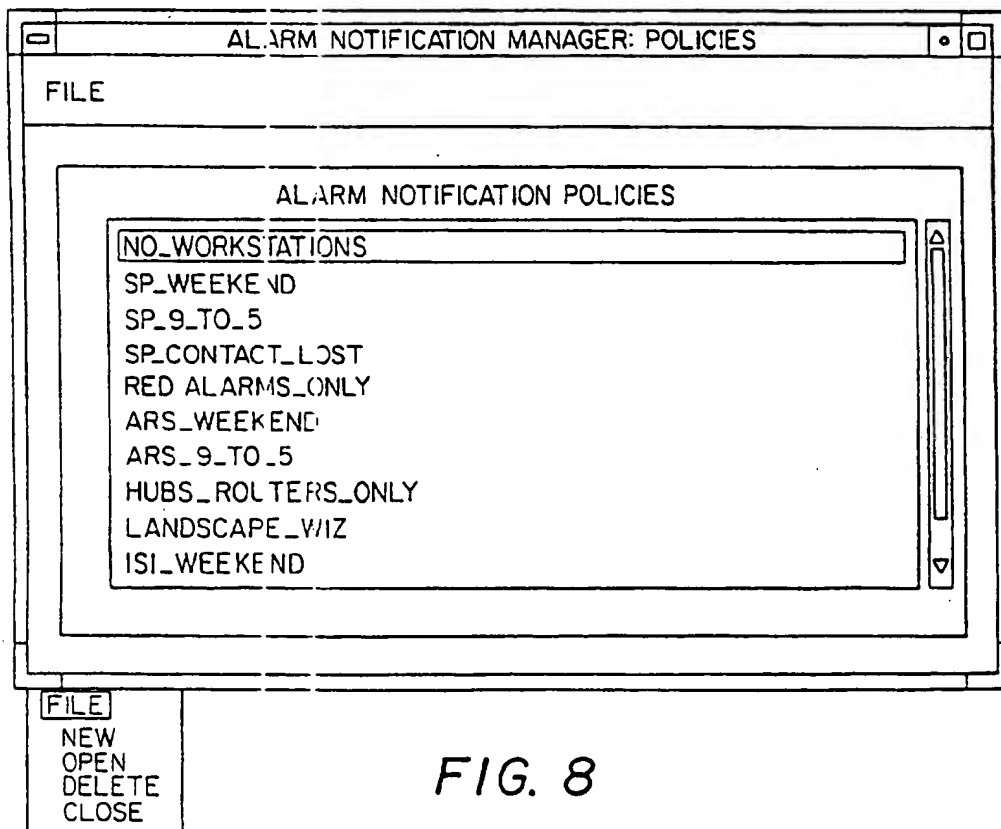


FIG. 8

SCHEDULER 17:11

SCHEDULE A TASK

SCHEDULED ENTRIES

WEEKLY ON MON AT 08:00 "ASSOCIATE APP=SPECTROPHONE"
WEEKLY ON FRI AT 17:00 "ASSOCIATE APP=SPECTROPHONE"

SCHEDULE

COMMAND

FREQUENCY

REPEAT ON:

☐ SUN ☐ MON ☐ TUE ☐ WED ☐ THU ☐ FRI ☐ SAT

AT

ADD **MODIFY** **REMOVE**

OK **APPLY** **RESET** **CANCEL**

FIG. 7

ALARM NOTIFICATION MANAGER: OPEN POLICY

POLICY: SP_WEEKEND

FILE VIEW

CREATE FILTER DUPLICATE FILTER DELETE NEGATE

FILTER 1

PARAMETER

AGE 02:30

TAG LEO LANFIXER

LANDSCAPE

WIZ
BRAT
ALTON
HOUND
CAPITAL

AND

MODEL TYPE

RTR_CISCO
RTR_WELLFLEET
RTR_IBM_WS
RTR_PROTEON

AND

IP SUBNET

134.141.141.0
134.141.143.0
134.141.65.0

AND

FILTER 2

PARAMETER

AGE 00:05

TAG WENDY WANWORKER

MODEL NAME

SCMCISCO.1-2
SCMCISCO.2-2
134.141.67.194
MIB-11

AND

SEVERITY

NOT YELLOW

OR

OR

FILE VIEW
EDIT
SAVE
SAVE AS
PRINT
CLOSE

FIG. 9

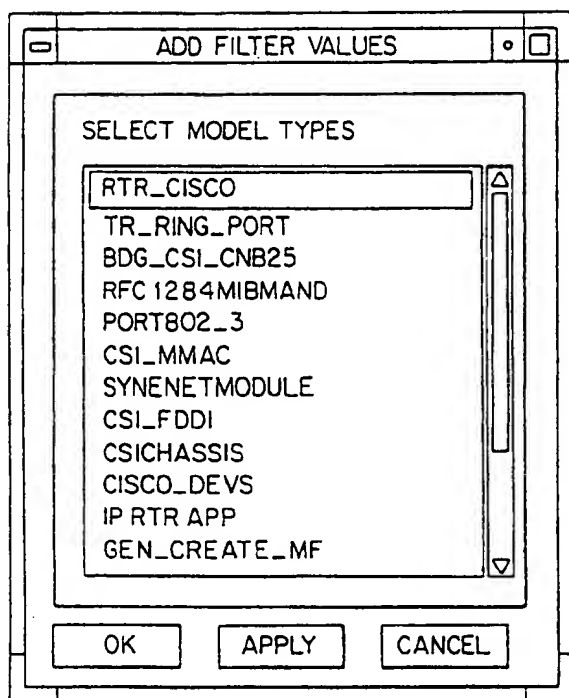


FIG. 10

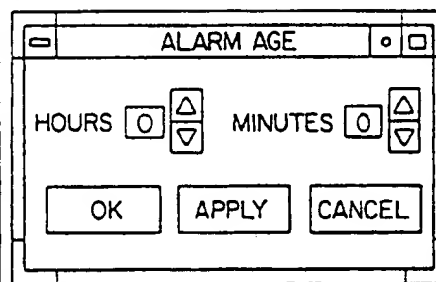


FIG. 11

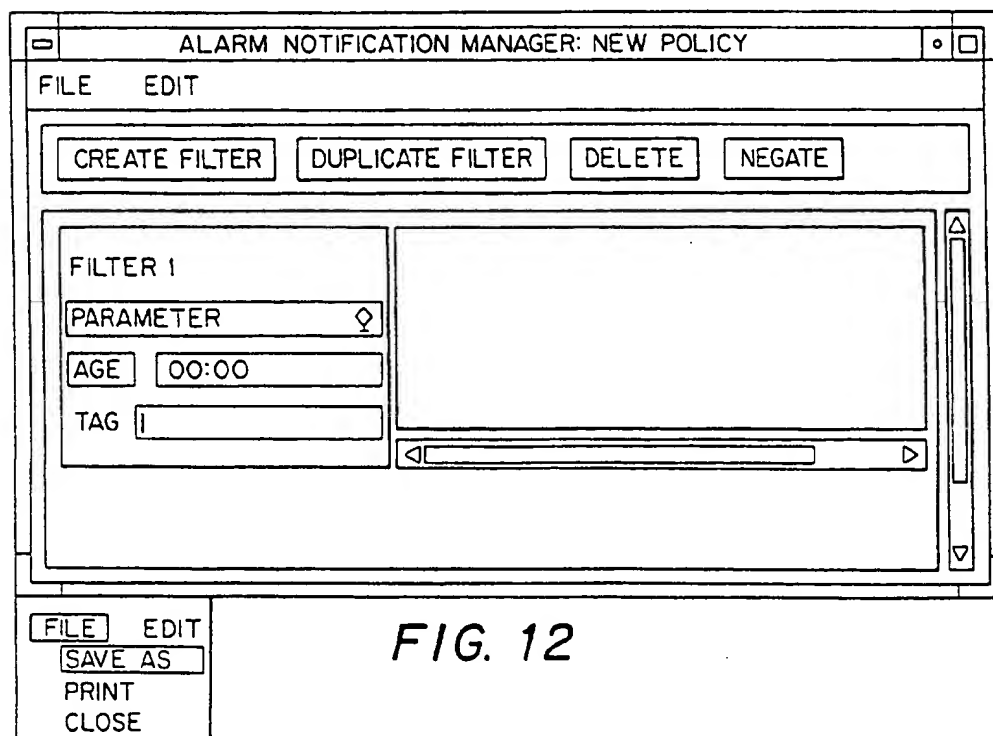


FIG. 12

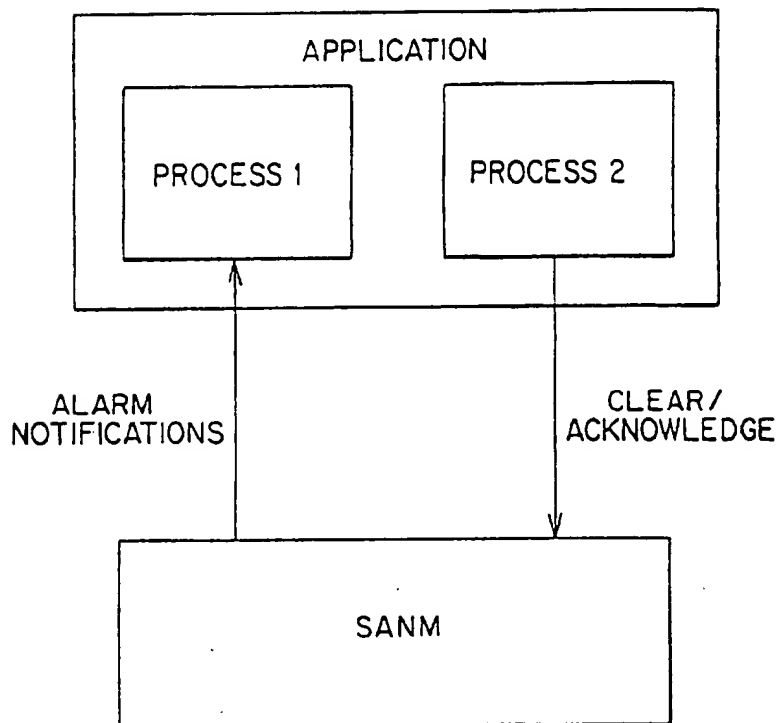


FIG. 13

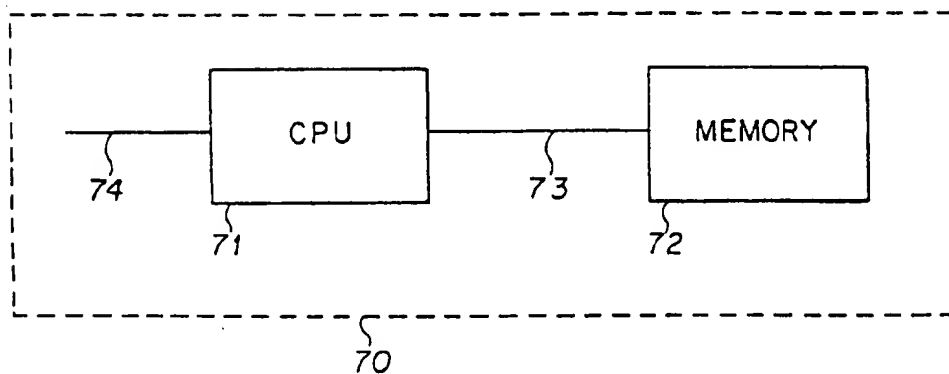


FIG. 14

METHOD AND APPARATUS FOR POLICY-BASED ALARM NOTIFICATION IN A DISTRIBUTED NETWORK MANAGEMENT ENVIRONMENT

FIELD OF THE INVENTION

The present invention relates to alarm notification in a communications network and more specifically to a method and apparatus for receiving alarms from multiple network management servers, applying policies to those alarms and forwarding the alarms that conform to the policies to one or more network management applications.

BACKGROUND OF THE INVENTION

Spectrum™ is a model-based network management system, sold by Cabletron Systems, Inc., Rochester, N.H. for maintaining and processing information pertaining to the condition of a communications network and providing the same to a user. For example, Spectrum™ will periodically poll a network device to request information, such as the number of packets sent on the network in a given time and the number of errors that occurred. If the error rate is above a predetermined limit, an error alarm is logged in the Spectrum™ database, an alarm sent to the user interface to notify the network manager, and a message is sent to shut off the corresponding network device.

Alternatively, if no response was received from the network device when it was polled, the reason for the loss of contact should be determined so that appropriate action, such as a service call, can be taken. In a network environment, loss of contact with a network device may be due to failure of that network device or to failure of another network device that is involved in the transmission of a message.

In many prior art network management systems, the network administrator was typically provided with a list of possible causes of a fault and was required to isolate the fault based on his experience and knowledge of the network. In Spectrum™, the system itself isolates network defaults using a technique known as Status Suppression. Spectrum™ maintains a database of models for each network device. When contact between a model and its corresponding network device is lost, the model sets a fault status and initiates the fault isolation technique. The model (first model) which lost contact with its corresponding network device (first network device) determines whether adjacent models have lost contact with their corresponding network devices; adjacent network devices are defined as those which are directly connected to a specified network device. If adjacent models cannot contact the corresponding network devices, then the first network device cannot be the cause of the fault, and its fault status in the first model will be overridden. By suppressing the fault status of the network devices which are determined not to be defective, the defective network device can be identified. Once the fault has been isolated, the condition of the defective device can be updated in the Spectrum™ database, a control message can be sent shutting off the defective device, and the network administrator can be notified via the user interface.

Spectrum™'s associated SpectroGRAPH™ user interface provides a graphical view into the network models. An alarm log view, shown in FIG. 1, includes an area 120 for the listing of current alarms, and an area 122 for displaying information pertaining to a selected alarm. The user may click on a particular alarm in the listing of current alarms to obtain more information. A multi-function icon 124 repre-

senting the network device having a fault is displayed in area 122, with one or more text fields 126 and 128 which provide information to the user regarding the cause of the alarm and the status of the device. By clicking on specified areas of the icon 124, the user can obtain further information regarding the device for which an alarm is registered.

Another method for fault management in large communications networks is to use a so-called "trouble-ticketing" system. This system provides a number of tools that can be used by network users, administrators, and repair and maintenance personnel. The basic data structure, a "trouble-ticket", has a number of fields in which a user can enter data describing the parameters of an observed network fault. A trouble-ticket filled out by a user may then be transmitted by, for example, an electronic mail system to maintenance and repair personnel. A trouble-ticket describing a current network fault that needs to be acted on is called "an outstanding trouble-ticket". When the network fault has been corrected, the solution to the problem, typically called a "resolution" is entered into an appropriate data field in the trouble-ticket and the trouble-ticket is said to be completed. The system provides for storage of completed trouble-tickets in memory and thus a library of such tickets is created, allowing users, administrators, and maintenance and repair personnel to refer to the stored completed trouble-tickets for assistance in determining solutions to future network faults. An example of a trouble-ticketing system is the ACTION REQUEST system, developed by Remedy Corporation, Mountain View, Calif., and sold by Cabletron Systems, Inc., Rochester, N.H.

ARS Gateway™ is a network management application sold by Cabletron Systems, Inc. which receives fault information from the Spectrum™ system and automatically generates a trouble-ticket that may be processed by the ACTION REQUEST system. This system is further described in copending and commonly owned U.S. Ser. No. 08/023,972 filed Feb. 26, 1993 by Lundy Lewis, and entitled "Method and Apparatus For Resolving Faults In Communications Networks," and which is hereby incorporated by reference in its entirety.

The Spectrum™ system is described in U.S. Pat. No. 5,261,044 issued Nov. 9, 1993 to Roger Dev et al., which is hereby incorporated by reference in its entirety. The Spectrum™ network management system is commercially available and also described in various user manuals and literature available from Cabletron Systems, Inc., Rochester, N.H.

Other network management platforms and applications for the basic filtering of alarms which are commercially available include: (1) HP OpenView, 3000 Hanover Street, Palto, Calif. 94304; (2) LattisNet, SynOptics Communications, 4401 Great American Pkwy., Santa Clara, Calif. 95054; (3) IBM Netview/6000, IBM Corp., Old Orchard Road, Armonk, N.Y. 10504; and (4) SunNet Manager, SunConnect, 2550 Garcia Ave, Mountain View, Calif. 94043.

Unfortunately, in the prior art systems alarms can only be received from one network management server. Also there is no provision for applying the same policy-based filter to multiple network management applications.

Thus, it is an object of the present invention to provide greater control over which alarms get reported to network management applications and to provide a means to ensure consistency of reported alarms across multiple network management applications.

SUMMARY OF THE INVENTION

The present invention is directed to an apparatus and method of alarm notification which includes: (a) receiving

3

alarms from multiple network management servers; (b) assigning policy-based filters to associated network management applications; and (c) applying the assigned policy-based filters to the alarms and for the alarms that pass the filters, generating an alarm notification forwarding the same to the associated network management applications.

In an embodiment described herein, a user designates a plurality of such filters, which constitute an alarm notification policy, to one or more associated network management applications. The policy-based filters are stored in a database, and a tag is assigned for identifying each filter. The same filters may be assigned to multiple applications.

In a further embodiment, the user may schedule the assignment of such policy-based filters to occur at a designated time in the future. For example, a user may pick a policy from a list of available policies to associate with a selected application, and then designate the frequency with which the policy is applied, e.g., once, hourly, daily, weekly or monthly.

Furthermore, the invention can be used in the same mode as similar tools in the prior art, i.e., with one alarm-forwarding component for each network management system/network management application pair, or alternatively as a single entity in a distributed network management environment.

These and other features of the present invention will be more fully described in the following detailed description and figures.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an example of an alarm log display provided by the prior art Spectrum™ network management system.

FIG. 2 is a block diagram of an alarm notification manager in accordance with the present invention, in use with multiple network management servers and multiple network management applications.

FIG. 3 is a flow chart illustrating the application of policy-based filters to an alarm, and forwarding of the alarm which passes the filters to an application in accordance with this invention.

FIG. 4 is an example of an Associations window display of the alarm notification manager.

FIG. 5 is an example of a New Association window display of the alarm notification manager.

FIG. 6 is an example of a Modified Association window display for the alarm notification manager.

FIG. 7 is an example of a Scheduler window display for the alarm notification manager.

FIG. 8 is an example of a Policies window display for the alarm notification manager.

FIG. 9 is an example of an Open Policy window display for the alarm notification manager.

FIG. 10 is an example of an Add Filter Values window display for the alarm notification manager.

FIG. 11 is an example of an Alarm Age window display for the alarm notification manager.

FIG. 12 is an example of a New Policy window display for the alarm notification manager.

FIG. 13 is a block diagram illustrating two separate processes between the network management application and the alarm notification manager.

FIG. 14 is a block diagram illustrating a central processing unit and memory for use in this invention.

DETAILED DESCRIPTION

The present invention is directed to an alarm notification manager which receives alarms from multiple network man-

4

agement servers, allows an unlimited number of filters to be defined within one policy, allows policies to be named and stored in a database, allows policies to be scheduled for different times, and allows the same policy to be applied to one or more network management applications.

As illustrated in FIG. 2, a live network 10 is connected by links 11 to one or more network management servers 12 which monitor the network. The servers detect errors or faults on the network and send alarm information to the alarm notification manager 14 via links 13. The alarm notification manager includes a policy database 16, method for applying policies to alarms 18, graphical interface 20, and scheduler 22. The manager applies policy-based filters to the alarm messages received from the servers, and for those alarms which pass the filter criteria, an alarm message is sent to the appropriate network management application 24 via links 23.

In a specific embodiment described herein, a plurality of distributed SpectroServers™, part of the Spectrum™ system sold by Cabletron Systems, Inc., Rochester, N.H., are used to model the live network 10, and several Spectrum™ applications receive the filtered alarm messages from the manager 14. These components have been implemented in the object-oriented programming language C++. However, the invention is not tied to any particular language nor to any particular products used in network management.

The Spectrum™ Network Management System

An understanding of the present invention is furthered by an understanding of the model-based network management system known as Spectrum™, which is described in U.S. Pat. No. 5,261,044, issued Nov. 9, 1993 to R. Dev et al., and hereby incorporated by reference in its entirety. The Spectrum™ network management system is commercially available and also described in various user manuals and literature available from Cabletron Systems, Inc., Rochester, N.H.

In summary, Spectrum™ is a system for maintaining and processing information pertaining to the condition of the computer network and providing the same to a user, the network including a plurality of network entities such as computer devices and software applications being executed on such devices. The system includes a virtual network machine, comprising a programmed digital computer, wherein a program is implemented using an object-oriented programming language such as C++, Eiffel, SmallTalk, and Ada. The virtual network consists of interrelated intelligent models of network entities and relations between network entities, including means for acquiring network data pertaining to the condition of a network entity from the corresponding network entity. The virtual network further includes means for maintaining objects which include network data relating to the corresponding network entity and one or more inference handlers for processing the network data, the inference handlers being responsive to changes occurring in the same and/or a different object. The network data can then be transferred to a user interface coupled to the virtual network machine, for supplying the network data to a user.

Thus, the models are implemented as software "objects" containing both "data" (attributes) relating to the corresponding network entity and one or more "inference handlers" (functions) for processing the data. See Grady Booch, "Object-Oriented Analysis And Design, With Applications," 2nd Edition, Benjamin/Cummings Publishing Co., Redwood City, Calif., Chapter 2, 1994. The inference handlers are initiated by predetermined virtual network events, such as a change in specified network data in the same model, a change in specified network data in a different model, and

predefined events or changes in models or model relations. Information pertaining to the condition of the network entity can be obtained from the network entity by polling the same, can be automatically received from the network entity (without polling), or can be inferred from data contained in other models. An alarm condition may be generated when the network data meets a predetermined criteria. Events, alarms and statistical information from the virtual network are stored in a database and are selectively displayed for the user.

The data in the Spectrum™ database may be used for generating topological displays of the network, showing hierarchical relationships between network devices, isolating a network fault, and reviewing statistical information.

Spectrum™ allows for collective management of autonomous local area networks (LANs), with equipment from different vendors. It complies with the current Simple Network Management Protocol (SNMP) standards, and can also accommodate other standard and proprietary protocols. The virtual network machine preprocesses the raw information coming from the network devices in order to construct a model of the network's current status and performance characteristics. Network elements that cannot be directly communicated with (e.g., cables and buildings) can infer their status from the status of the devices connected to (or contained within) them. The virtual network machine provides a consistent interface for management applications to access any of the information in the model and thereby provides these applications with a unified view of the network.

Spectrum's™ associated SpectroGRAPH™ user interface provides a highly graphical multi-perspective view into the network model. SpectroGRAPH™ enables the user to navigate through a landscape in which cables, networks, local area networks and even rooms show up as icons, and which icons indicate the health and performance characteristics of those elements. These icons can be further queried for additional information. SpectroGRAPH™'s main function is to visually present to the user the model within the virtual network machine. It allows the user to navigate freely within the network model, only limited by the access rights assigned by the network administrator. The information can be accessed at varying degrees of detail, from a macro overview, to the devices and cables which connect them. In addition to its navigation functions, SpectroGRAPH™ provides an alarm management facility, an event log window, a reporting facility, a find facility, and other features.

The above description of the Spectrum™ system provides a context for an understanding of the present invention.

The Alarm Notification Manager

The following definitions are helpful to an understanding of the present invention:

SANM

SPECTRUM™ Alarm Notification Manager

Policy

A set of criteria which a given alarm must satisfy in order to be passed to the application with which the policy is associated. A policy may consist of one or more filters.

Filter

A set of filter parameters and associated filter values. Multiple filters define multiple sets of values for the filter parameters.

Filter Parameter

A data type such as model name or IP subnet for which the user can specify a value or list of values. SANM provides the user with a fixed list of filter parameters.

Association

When the user associates a policy with an application, he is specifying the filter criteria that SANM should apply to the alarms it sends to the application.

A filter consists of a list of filter parameters and a list of associated filter values. A user (of a network management application) specifies the value(s) that each filter parameter can take in order for a given alarm to pass the filter criteria. The following is a list of representative filter parameters:

- model name
- model type name
- device IP subnet
- device location
- alarm severity
- alarm age
- SpectroSERVER host name
- landscape name
- alarm cause

The value for each of the above filter parameters would be received from Spectrum™, except for the alarm age parameter. The alarm age parameter is used internally by SANM and specifies the length of time that it should hold an alarm before sending it to an application. If the alarm is cleared by Spectrum™ during this time, it is not sent to the application. This feature may be used to filter out transient alarms.

Each filter value also has a corresponding flag which indicates whether it should be negated. For example, if the negate flag is set for a model type name value of Hub_CSI_IRM3, this filter value states that all alarms for models NOT of type Hub_CSI_IRM3 should pass.

More complex filtering can be achieved by defining multiple filters within a policy. Each filter specifies a separate set of filter values.

SANM performs a logical AND of all the filter criteria within a filter and performs a logical OR between all filters within a policy.

For example, a policy contains two filters as follows:

Filter 1

Model Type: Rtr_Cisco
Landscape: wiz

Filter 2

Model Type: Rtr_Wellfleet
Landscape: brat

SANM would apply this policy to a given alarm as follows:

IF the alarm has:

model type Rtr_Cisco AND landscape wiz

OR

model type Rtr_Wellfleet AND landscape brat

THEN send the alarm to the application.

Each filter also contains a filter tag, which is a text string that the user enters. This tag, which is included in the alarm notification, identifies which filter(s) passed and can be used by an application to perform routing of alarms.

For example, a different user name can be entered in the filter tag field of each filter, so that if the criteria in one filter pass, the application will notify a particular user, whereas if the criteria in another filter pass, the application will notify a different user. If multiple filters pass, a list of corresponding filter tags is sent in the alarm notification.

Another example of the SANM filtering mechanism is shown in FIG. 3. In this figure, the criteria listed within each filter are the criteria for which values have been specified by the user. It can be seen from this example that all filters are applied in parallel to a given alarm (i.e., a logical OR is performed between filters). However, all criteria within a

given filter must be satisfied for the alarm to pass the filter (i.e., a logical AND is performed between the criteria within a given filter). Since, in this example, the alarm passes the criteria in filters 1 and 3, an alarm notification containing filter tags "A" and "C" is sent to the application.

Policies and the associations between policies and applications are stored in the SPECTRUM™ database. This means that the same policies are available to any client machine running SANM. It also means that the policy names contained in event messages logged by SANM have significance to all client machines using the same SPECTRUM™ database.

1.0 Alarm Notification

After an application has registered with SANM to receive alarms, an alarm notification is sent to that application each time an alarm is received from SPECTRUM™ that passes the criteria specified in the policy associated with that application. The information contained in each alarm notification consists of the real-time values of each filter parameters, plus the values of the following parameters:

- model handle
- model type handle
- model condition value
- model security string
- alarm ID
- alarm time
- alarm probable cause
- alarm status
- event message associated with alarm
- assigned repair person
- user-clearable flag

One exception to this is that an IP subnet address may be specified as a filter criterion, but the full IP address of the device that created the alarm is passed in the alarm notification.

A notification that an alarm has been cleared or updated is sent to an application when SANM receives such a notification from SPECTRUM™, but only if the alarm which is being cleared or updated was initially sent to the application when it occurred (i.e., it passed the filter criteria for that application).

2.0 Configuration Tool

The SANM Configuration Tool enables the user to define Alarm Notification Policies and to associate these policies with the applications that use SANM.

The Configuration Tool is invoked by selecting Alarm Notification Manager from the asterisk menu of SpectroGRAPH™.

2.1 Associations Window

When the Configuration Tool is invoked, the first window to appear is the Associations window, shown in FIG. 4. This window displays a list of the currently defined SANM applications and the policy that is associated with each of them.

A new association is created by selecting New from the File menu. This brings up the New Association window shown in FIG. 5.

An existing association is modified by selecting the association and then selecting Modify from the File menu. This brings up the Modify Association window shown in FIG. 6.

An existing association is deleted by selecting the association and then selecting Delete from the File menu. The selected association is deleted after the user confirms the operation in a Confirmation Dialog window (not shown).

The modification of an existing association can be scheduled by selecting the association and then selecting Schedule from the File menu. This brings up the Scheduler window shown in FIG. 7.

All currently defined policies can be viewed by selecting Policies from the Tools menu. This brings up the Policies window shown in FIG. 8.

2.2 New Association Window

The New Association Window is illustrated in FIG. 5. In this window, a policy is selected from the list of available policies and the application name is entered. When OK is pressed, the window disappears and the new association appears in the Associations window (FIG. 4).

2.3 Modify Association Window

The Modify Association window is illustrated in FIG. 6. In this window, the user picks a policy from the list of available policies to associate with the selected application (SpectroPHONE™ in this example, available from Cabletron Systems, Inc.). Pressing OK makes this window disappear and the modified association is displayed in the Associations window (FIG. 4).

2.4 Scheduler Window

The Scheduler window is illustrated in FIG. 7. Pressing the Associate button brings up the Modify Association window illustrated in FIG. 6. In the Modify Association window, the user picks a policy from the list of available policies to associate with the selected application (SpectroPHONE™ in this example). In the Scheduler window, the user then presses the Frequency button to specify the frequency of the association. The Frequency options are: Once, Hourly, Daily, Weekly and Monthly. The information in the area below the Frequency button changes depending on what frequency option is selected as follows:

The Once option allows the user to specify the month, day and start-time.

The Hourly option allows the user to specify the number of minutes after each hour.

The Daily option allows the user to specify the time.

The Weekly option allows the user to specify the day of the week and the time.

The Monthly option allows the user to specify the day of the month and the time.

Once the desired scheduling options have been selected, pressing the Add button inserts the scheduling information into the Scheduled Entries portion of the window. Further entries can be added by repeating the previous steps. Entries can be modified and removed by selecting them and using the Modify and Remove buttons.

2.5 Policies Window

The Policies Window is illustrated in FIG. 8. This window shows all currently defined policies.

A new policy is created by selecting New from the File menu. This causes the New Policy window (FIG. 12) to appear.

An existing policy is viewed and modified by selecting the policy and then selecting Open from the File menu. This causes the open Policy window (FIG. 9) to appear.

An existing policy is deleted by selecting the policy and then selecting Delete from the File menu. The selected policy is deleted after the user confirms the operation in a Confirmation Dialog window (not shown).

2.6 Open Policy Window

The Open Policy window is illustrated in FIG. 9. This window shows all the filters that make up the policy. In the example shown in FIG. 9, Filters 1 and 2 are visible, but subsequent filters can be viewed using the scrollbar on the

right of the window. Similarly, the other filter parameters for Filter 1 and their associated values can be viewed using the scrollbar below the Filter 1 filter parameters.

To modify the displayed policy, Edit must be selected from the File menu. The View item in the menu bar then becomes Edit. Once in Edit mode, multiple values for a particular filter parameter can be deleted or negated by selecting the values and pressing the Delete or Negate button. Values can be added for a particular filter parameter by pressing the filter parameter button (e.g. Landscape or Model Type). This brings up a separate window containing a list of available values from which multiple values can be selected. An example of this window is shown in FIG. 10.

Filter parameters may be added to a filter by pressing the Parameter button within the filter. A pop-up menu appears containing all eight filter parameters. However, those filter parameters which are already present in the filter are greyed-out and cannot be selected. Selecting one of the available filter parameters from this menu causes the new filter parameter and associated value box to appear in the filter.

The alarm age for a particular filter can be modified by pressing the Age button in the Open Policy window. This brings up the Alarm Age window shown in FIG. 11. The values for the Hours and Minutes fields initially contain the values from the Age text field in the Open Policy window. These values can be modified using the up and down arrow buttons for hours and minutes.

A filter tag can be modified in the Open Policy window by typing directly into the Tag text field of a filter.

A new filter may be added to the policy displayed in the Open Policy window by pressing the Create Filter button. This will cause a new filter with no filter parameters to be added to the end of the list of filters.

An existing filter may also be duplicated. To do this the filter to be duplicated must first be selected by clicking within the filter label field (e.g. the area around the label Filter 2) and then pressing the Duplicate Filter button. Doing this causes a new filter, containing the same filter parameters and values as the selected filter, to be added to the end of the filter list. This new filter can then be modified.

After modifying a policy, Save can be selected from the File menu to save the modified policy under its existing name, or Save As can be selected to save the modified policy under a different name.

The information in the Open Policy window can be printed by selecting Print from the File menu.

2.7 New Policy Window

The New Policy Window is illustrated in FIG. 12. The operations that can be performed in the New Policy window are the same as those performed in the Open Policy window (FIG. 9). No filter parameters initially appear within Filter 1, therefore the first operation that needs to be performed is to select a filter parameter by pressing the Parameter button within Filter 1. All filter parameters are available from the pop-up menu at this point because the filter does not yet contain any filter parameters.

A new policy is saved by selecting Save As from the File menu and entering the name for the policy in a dialog box.

3.0 Integration of SANM and Application

A developer would use the following interface to integrate an application written in C or C++ with the Spectrum™ alarm mechanism.

An application using SANM to receive alarm notifications and to clear/acknowledge alarms requires two separate processes, as illustrated in FIG. 13.

As an example of how these two separate processes would be used in an application, the ARS Gateway™ product

would use Process 1 to receive filtered alarms from SANM, format them into Trouble Tickets and put them into the ARS Database. Process 2 would be used when a user viewing a particular Trouble Ticket pressed a clear or acknowledge button in the Trouble Ticket.

Two different programming paradigms are required for the two application processes that use SANM:

For the process that receives alarm notifications from SANM, an asynchronous callback paradigm is used. This means that when the application code registers with SANM to receive alarms, it hands program control over to SANM. When SANM needs to send an alarm notification to the application, the application receives a callback from SANM. This process is terminated by sending it a TERM (terminate, 15) signal.

For the process that clears or acknowledges alarms, however, a synchronous paradigm is used. This means that the application code in this process has program control. When this application code makes a call to the SANM API to clear or acknowledge an alarm, the call blocks the application until it is finished.

3.1 Definitions and Data Structures

All definitions and data structures are contained in the SANM header file sanm.h and are described below.

The prototype for the application's callback functions is defined as follows: typedef void (*SANMcb) (struct SANM_Alarm_Notify *);

All the data in an alarm notification is contained in the SANM_Alarm_Notify structure, which is defined as follows:

```

struct SANM_Alarm_Notify{
    char          *model_name;
    SANMulong     model_handle;
    char          *model_type_name;
    SANMulong     model_type_handle;
    int           condition_value;
    char          *security_string;
    SANMulong     alarm_ID;
    SANMTimestamp alarm_time;
    SANMulong     cause_code;
    char          *probable_cause;
    char          *alarm_status;
    char          *event_message;
    char          *repair_person;
    char          *IP_address;
    char          *location;
    SANMulong     severity;
    SANMulong     alarm_age;
    char          *SpectroSERVER_host;
    char          *landscape;
    SANMBoolean   user_clearable;
    char          *filter_tag;
};

```

All errors and warnings are defined in the enumeration SANM_error as follows:

```

enum SANM_error
{
    SANM_RETURN_OK,
    SANM_INVALID_ALARM,
    SANM_INVALID_LANDSCAPE,
    SANM_ALARM_NOT_CLEARABLE,
    SANM_REGISTER_ERROR }

```

3.2 Functions

The functions that make up the SANM C/C++ API are described in the following sections in manual page format.

3.2.1 SANMInit

NAME

SANMInit—initialize interaction with SANM

SYNOPSIS

#include "sanm.h"

SANM_error SANMInit (char *application_name,
SANMBoolean rcv_or_clr);

DESCRIPTION

SANMInit serves to initialize the program for interaction with SANM. This function should be called from within both application processes before any other function in the SANM API.

INPUT ARGUMENTS

application_name

the name which must be used by the user to identify this application when using the Configuration Tool to associate a policy with it.

rev_or_clr

a flag which indicates whether this process is going to receive alarm notifications or clear/acknowledge alarms. The flag can take either of the following two values:

SANM_RCV_ALARMS
SANM_CLR_ALARMS

RETURN VALUES

status

The return value will be one of the following values:
SANM_RETURN_OK

3.2.2 SANMRegister

NAME

SANMRegister—register with SANM

SYNOPSIS

#include "sanm.h"

SANM_error SANMRegister (SANMCb set_cb,
SANMCb clear_cb,
SANMCb update_cb);

DESCRIPTION

SANMRegister registers the application to receive alarm notifications from SANM. By calling this function, the application hands program control over to SANM until one of the application's callback functions is called.

INPUT ARGUMENTS

set_cb

the name of the function that SANM will call in order to send an alarm notification for a new alarm. All applications must pass a valid function for this parameter.

clear_cb

the name of the function that SANM will call in order to send an alarm notification for a cleared alarm. This parameter can be NULL if the application does not want to receive notifications for cleared alarms.

update_cb

the name of the function that SANM will call in order to send an alarm notification for an updated alarm. This parameter can be NULL if the application does not want to receive notifications for updated alarms.

RETURN VALUES

status

In normal operation, this function will never return. However, if it fails, one of the following errors will be returned:

SANM_REGISTER_ERROR

3.2.3 SANMClear

NAME

SANMClear—clear an alarm

SYNOPSIS

#include "sanm.h"

SANM_error SANMClear (SANMulong alarm_ID,
char *landscape);

DESCRIPTION

SANMClear clears an alarm in SPECTRUM. An application can only clear alarms for which it received notifications from SANM. Also, the user_clearable flag must have been set to CLEARABLE in the alarm notification

INPUT ARGUMENTS

alarm-ID

the ID of the alarm to be cleared

landscape

the landscape that generated the alarm

RETURN VALUES

status

The return value will be one of the following values:

SANM_RETURN_OK
SANM_INVALID_ALARM
SANM_INVALID_LANDSCAPE
SANM_ALARM_NOT_CLEARABLE

3.2.4 SANMack

NAME

SANMack—acknowledge an alarm

SYNOPSIS

#include "sanm.h"

SANM_error SANMack (SANMulong alarm_ID,
char *landscape);

DESCRIPTION

SANMack acknowledges an alarm in SPECTRUM. An application can only acknowledge alarms for which it received notifications from SANM.

INPUT ARGUMENTS

alarm_ID

the ID of the alarm to be acknowledged

landscape

the landscape that generated the alarm

RETURN VALUES

status

The return value will be one of the following values:

SANM_RETURN_OK
SANM_INVALID_ALARM
SANM_INVALID_LANDSCAPE

The present embodiments may be implemented in a general purpose computer 70 as shown in FIG. 14. The general purpose computer may include a computer processing unit (CPU) 71, memory 72, a processing bus 73 by which the CPU can access the memory, and interface 74 to the rest of the alarm notification manager.

In alternative embodiments, the invention may be a computer apparatus which performs the functions of any of the previous embodiments. Alternatively, the invention may be a memory, such as a floppy disk, compact disk, or hard drive, that contains the computer program or data structure, for providing to a general purpose computer instructions and data for carrying out the functions of the previous embodiment.

Having thus described certain particular embodiments of the invention, various modifications will readily occur to those skilled in the art which are intended to be within the scope of this invention. Accordingly, the foregoing description is by way of example only, and not intended to be limiting.

13

What is claimed is:

1. A method of alarm notification comprising the steps of:

- (a) receiving alarms from multiple network management servers;
- (b) creating, in response to a user action, a plurality of policy-based filters; and
- (c) applying the policy-based filters to the received alarms, generating an alarm notification and forwarding the same to at least one network management application.

2. The method of claim 1, wherein:

the creating step includes creating a plurality of filters comprising a policy.

3. The method of claim 2, wherein:

each filter comprises at least one filter parameter; and the applying step comprises performing a logical AND of all parameters within one filter and performing a logical OR between all filters within one policy.

4. The method of claim 3, wherein:

the generating step includes specifying real-time values of each filter parameter in the alarm notification.

5. The method of claim 2, wherein:

the creating step includes storing a policy name and at least one associated network management application in a database accessible to all servers.

6. The method of claim 1, wherein:

the creating step includes assigning a tag to each filter.

7. The method of claim 6, wherein:

the generating step includes specifying the tag for the filter which the alarm passed in the alarm notification.

14

8. The method of claim 2, wherein:

the creating step includes storing the filters in a database.

9. The method of claim 2, wherein:

the generating step further includes specifying a user name in the alarm notification to enable an application which receives the alarm notification to notify a user having the specified user name.

10. The method of claim 2, wherein:

the creating step includes scheduling the creating to occur at a specified time.

11. The method of claim 2, further comprising:

(d) following resolution of an alarm, forwarding an alarm clear message to the at least one network management application.

12. The method of claim 2, wherein:

the creating step includes assigning the same filters to multiple network management applications.

13. The method of claim 2, wherein:

the creating step is performed by a user via a graphical user interface.

14. The method of claim 2, wherein:

the generating step includes generating an alarm notification which contains information about the device which generated the alarm.

15. The method of claim 2, further comprising:

the at least one network management application generating an alarm clear message and forwarding the same to the network management server which generated the alarm.

* * * * *



US006088356A

United States Patent [19]

Hendel et al.

[11] Patent Number: **6,088,356**
 [45] Date of Patent: **Jul. 11, 2000**

- [54] **SYSTEM AND METHOD FOR A MULTI-LAYER NETWORK ELEMENT**
- [75] Inventors: Ariel Hendel, Cupertino; Leo A. Hejza; Shree Murthy, both of Sunnyvale; Louise Yeung, San Carlos, all of Calif.
- [73] Assignee: Sun Microsystems, Inc., Mountain View, Calif.
- [21] Appl. No.: 08/884,244
- [22] Filed: Jun. 30, 1997
- [51] Int. Cl.⁷ H04L 12/56
- [52] U.S. Cl. 370/392; 370/401; 370/469
- [58] Field of Search 370/230, 252, 370/389, 392, 399, 400, 401, 402, 406, 407, 408, 469, 471

[56] References Cited

U.S. PATENT DOCUMENTS

4,539,637	9/1985	DeBruler	364/200
4,627,052	12/1986	Hoare et al.	370/88
4,641,302	2/1987	Miller	370/60
4,652,874	3/1987	Loyer	340/825.05
4,737,953	4/1988	Koch et al.	370/94

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

13016 6/1998 WIPO.

OTHER PUBLICATIONS

"Foundry Products", downloaded from Website <http://www.foundrynet.com/> on Jun. 19, 1997.

Anthony J. McAuley & Paul Francis, "Fast Routing Table Lookup Using CAMs", IEEE, 1993, pp. 1382-1390.

"Gigabit Ethernet", Network Strategy Report, The Burton Group, v2, May 8, 1997 40 pages.

"IP On Speed", Erica Roberts, Internet-Draft, Data Communications on the Web, Mar. 1997, 12 pages.

"Multilayer Topology", White Paper, Internet-Draft, 13 pages, downloaded from Website <http://www.baynetworks.com> on Apr. 18, 1997.

International Search Report, PCT/US98/13368, 5 pages.

International Search Report, PCT/US98/13364, 4 pages.

International Search Report, PCT/US98/13365, 4 pages.

International Search Report, PCT/US98/13177, 4 pages.

International Search Report, PCT/US98/13199, 5 pages.

International Search Report, PCT/US98/13015, 5 pages.

International Search Report, PCT/US98/13202, 4 pages.

International Search Report, PCT/US98/13361, 5 pages.

International Search Report, PCT/US98/13200, 6 pages.

International Search Report, PCT/US98/13203, 7 pages.

International Search Report, PCT/US98/13206, 8 pages.

International Search Report, PCT/US98/13362, 5 pages.

Wang et al., A Novel Message Switch for Highly Parallel Systems, IEEE, pp. 150-155, 1989.

Tobagi, Fast Packet Switch Architectures for Broadband Integrated Services Digital Networks, Proceedings of the IEEE, vol. 78, Issue 1, pp. 133-167, Jan. 1990.

Fliesser et al., Design of a Multicast ATM Packet Switch, Electrical and Computer Engineering, 1993 Canadian Conference, pp. 779-783, 1993.

(List continued on next page.)

Primary Examiner—Chau Nguyen

Assistant Examiner—Soon-Dong Hyun

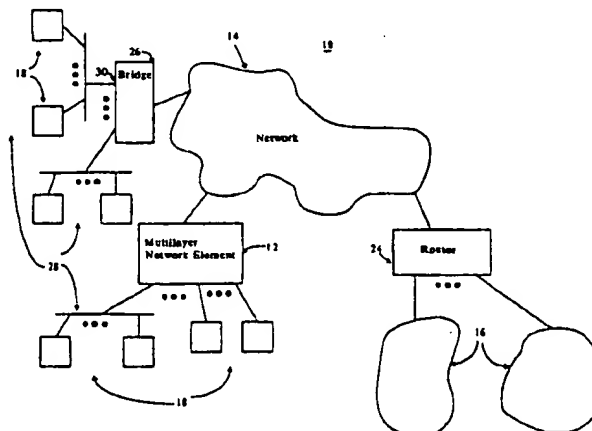
Attorney, Agent, or Firm—Blakely Sokoloff Taylor & Zafman

[57]

ABSTRACT

A multi-layer network element for forwarding received packets from an input port to one or more output ports. The packet is examined to look for first and second forwarding information. A packet is also assigned to a class and provided with default packet forwarding information. An associative memory is searched once for each type of information. The results from the two searches are combined with the default packet forwarding information to forward the packet to the appropriate one or more output ports. In some instances, the results of the first search dominate the forwarding decision, in other, the results of the second search dominate the forwarding decision, and in still other instances, the default information dominates.

23 Claims, 7 Drawing Sheets



U.S. PATENT DOCUMENTS

4,807,111	2/1989	Cohen et al.	364/200	5,583,981	12/1996	Pleyer	395/326
4,811,337	3/1989	Hart	370/85	5,592,476	1/1997	Calamvokis et al.	370/390
4,850,042	7/1989	Petronio et al.	455/606	5,594,727	1/1997	Kolbenson et al.	370/468
4,899,333	2/1990	Roediger	370/427	5,600,641	2/1997	Duault et al.	370/400
4,922,503	5/1990	Leone	370/85.13	5,602,841	2/1997	Lebizay et al.	370/413
4,933,938	6/1990	Sheehy	370/85.13	5,606,669	2/1997	Bertin et al.	395/200.15
4,935,869	6/1990	Yamamoto	364/200	5,608,726	3/1997	Virgile	370/401
5,130,977	7/1992	May et al.	370/60	5,610,905	3/1997	Murthy et al.	370/401
5,150,358	9/1992	Punj et al.	370/468	5,615,340	3/1997	Dai et al.	395/200.17
5,159,685	10/1992	Kung	395/575	5,617,421	4/1997	Chin et al.	370/402
5,163,046	11/1992	Hahne et al.	370/79	5,619,497	4/1997	Gallagher et al.	370/394
5,210,746	5/1993	Maher et al.	370/79	5,619,500	4/1997	Hiekali	370/414
5,220,562	6/1993	Takada et al.	370/401	5,619,661	4/1997	Crews et al.	395/299
5,231,633	7/1993	Hluchyj et al.	370/94.1	5,623,489	4/1997	Cotton et al.	370/381
5,251,205	10/1993	Callon et al.	370/60	5,633,710	5/1997	Mandal et al.	364/514
5,278,830	1/1994	Kudo	370/94.1	5,633,865	5/1997	Short	370/412
5,291,482	3/1994	McHarg et al.	370/413	5,636,371	6/1997	Yu	395/500
5,293,379	3/1994	Carr	370/474	5,640,605	6/1997	Johnson et al.	395/881
5,301,333	4/1994	Lee	395/725	5,649,109	7/1997	Griesmer et al.	395/200.17
5,309,437	5/1994	Perlman et al.	340/827	5,651,002	7/1997	Van Seters et al.	370/392
5,313,454	5/1994	Bustini et al.	370/13	5,675,741	10/1997	Aggarwal et al.	370/200.12
5,343,471	8/1994	Cassagnol	370/85.13	5,684,800	11/1997	Dobbins et al.	370/401
5,353,412	10/1994	Douglas et al.	395/325	5,689,506	11/1997	Chiussi et al.	370/388
5,365,514	11/1994	Hershey et al.	370/17	5,689,518	11/1997	Galand et al.	371/37.1
5,386,413	1/1995	McAuley et al.	370/54	5,691,984	11/1997	Gardner et al.	370/401
5,392,432	2/1995	Engelstad et al.	395/700	5,706,472	1/1998	Ruff et al.	395/497.04
5,394,402	2/1995	Ross	370/94.1	5,720,032	2/1998	Picazo, Jr. et al.	395/200.8
5,396,602	3/1995	Amini et al.	395/325	5,724,348	3/1998	Basso et al.	370/384
5,402,415	3/1995	Turner	370/60	5,724,358	3/1998	Headrick et al.	370/418
5,404,538	4/1995	Krappweis, Sr.	395/725	5,726,977	3/1998	Lee	370/235
5,410,540	4/1995	Aiki et al.	370/390	5,734,651	3/1998	Blakeley et al.	370/392
5,410,722	4/1995	Cornaby	395/800	5,734,865	3/1998	Yu	395/500
5,420,862	5/1995	Perlman	370/85.13	5,740,171	4/1998	Mazzola et al.	370/392
5,422,838	6/1995	Lin	365/49	5,740,175	4/1998	Wakeman et al.	395/422
5,425,026	6/1995	Mori	370/60	5,740,375	4/1998	Dunne et al.	395/200.68
5,425,028	6/1995	Britton et al.	370/94.1	5,742,604	4/1998	Edsall et al.	370/401
5,426,736	6/1995	Guineau, III	395/250	5,742,760	4/1998	Picazo, Jr. et al.	370/351
5,432,907	7/1995	Picazo, Jr. et al.	395/200	5,745,048	4/1998	Taguchi et al.	340/870.01
5,450,399	9/1995	Sugita	370/60.1	5,748,631	5/1998	Bergantino et al.	370/398
5,455,820	10/1995	Yamada	370/413	5,748,905	5/1998	Hauser et al.	395/200.79
5,457,681	10/1995	Gaddis et al.	370/402	5,751,967	5/1998	Raab et al.	395/200.58
5,459,714	10/1995	Lo et al.	370/13.1	5,751,971	5/1998	Dobbins et al.	395/200.68
5,459,717	10/1995	Mullan et al.	370/351	5,754,540	5/1998	Liu et al.	370/315
5,461,611	10/1995	Drake, Jr. et al.	370/54	5,754,774	5/1998	Bittinger et al.	395/200.33
5,461,624	10/1995	Mazzola	370/402	5,754,801	5/1998	Lambrecht et al.	395/308
5,473,607	12/1995	Hausman	370/85.13	5,757,771	5/1998	Li et al.	370/235
5,477,537	12/1995	Dankert et al.	370/60	5,757,795	5/1998	Schnell	370/392
5,481,540	1/1996	Huang	370/85.13	5,761,435	6/1998	Fukuda et al.	395/200.68
5,485,455	1/1996	Dobbins et al.	370/255	5,764,634	6/1998	Christensen et al.	370/401
5,485,578	1/1996	Sweazey	395/200.54	5,764,636	6/1998	Edsall	370/401
5,490,139	2/1996	Baker et al.	370/60	5,781,549	7/1998	Dai	370/398
5,490,252	2/1996	Macera et al.	395/200.01	5,784,559	7/1998	Frazier et al.	395/200.13
5,490,260	2/1996	Miller et al.	395/427	5,784,573	7/1998	Szczepanek et al.	395/200.8
5,493,564	2/1996	Mullan	370/351	5,790,546	8/1998	Dobbins et al.	370/400
5,500,860	3/1996	Perlman et al.	370/85.13	5,790,808	8/1998	Seaman	395/200.53
5,509,123	4/1996	Dobbins et al.	395/200.15	5,802,047	9/1998	Kinoshita	370/359
5,515,376	5/1996	Murthy et al.	340/402	5,802,052	9/1998	Venkataraman	370/395
5,517,488	5/1996	Miyazaki et al.	370/16	5,802,278	9/1998	Isfeld et al.	395/200.02
5,535,202	7/1996	Kondoh	370/60.1	5,812,527	9/1998	Kline et al.	370/232
5,550,816	8/1996	Hardwick et al.	370/60	5,815,737	9/1998	Buckland	370/395
5,553,067	9/1996	Walker et al.	370/60	5,822,319	10/1998	Nagami et al.	370/392
5,555,405	9/1996	Griesmaer et al.	395/600	5,825,767	10/1998	Mizukoshi et al.	370/395
5,557,610	9/1996	Calamvokis et al.	370/60.1	5,825,772	10/1998	Dobbins et al.	370/396
5,561,666	10/1996	Christensen et al.	370/434	5,835,491	11/1998	Davis et al.	370/386
5,561,791	10/1996	Mendelson et al.	395/550	5,838,677	11/1998	Kozaki et al.	370/389
5,563,878	10/1996	Blakeley et al.	370/392	5,838,681	11/1998	Bonomi et al.	370/395
5,566,170	10/1996	Bakke et al.	370/60	5,852,607	12/1998	Chin	370/401
5,570,365	10/1996	Yodhida	370/85.6	5,856,977	1/1999	Yang et al.	370/411
5,572,522	11/1996	Calamvokis et al.	370/395	5,859,849	1/1999	Parks	370/395
5,574,861	11/1996	Lorvig et al.	395/200.06	5,867,677	2/1999	Tsukamoto	395/311
				5,872,783	2/1999	Chin	370/392
				5,872,904	2/1999	McMillen et al.	395/182.02

5,875,464	2/1999	Kirk	711/129
5,878,043	3/1999	Casey	370/397
5,878,232	3/1999	Marimuthu	395/200.79
5,892,912	4/1999	Suzuki et al.	395/200.48
5,898,687	4/1999	Harriman et al.	370/390
5,931,980	11/1998	Varma et al.	370/395

OTHER PUBLICATIONS

Chang et al., An Overview of the Pipelined Common Buffer Architecture (PCBA) for Memory Based Packet/Cell Switching Systems, Local Computer Networks, 1994, pp. 288-297, 19th Conference, IEEE.

Agrawal et al., A Scalable Shared Buffer ATM Switch Architecture, VLSI, 1995 5th Great Lakes Symposium, IEEE, pp. 256-261, 1994.

Sabaa et al., Implementation of a Window-Based Scheduler in an ATM Switch, Electrical and Computer Engineering, 1995 Canadian Conference, IEEE, pp. 32-35, 1995.

Naraghi-Pour et al., A Multiple Shared Memory Switch, System Theory, 1996 Southeastern Symposium, IEEE, pp. 50-54, 1996.

Iyengar et al., Switching Prioritized Packets, Globecom'89: IEEE Global Telecommunications Conference, pp. 1181-1186, 1989.

Microsoft Press, "Microsoft Computer Dictionary Fourth Edition", Microsoft Corporation, 1999, 4 pages.

"Load Balancing for Multiple Interfaces for Transmission Control Protocol/Internet Protocol for VM/MVS", IBM Technical Disclosure Bulletin, 38(9): 7-9 (Sep., 1995).

T. Nishizono et al., "Analysis on a Multilink Packet Transmission System", Electron. Commun. JPN 1, Commun., (USA), 68(9): 98-104 (Sep., 1985).

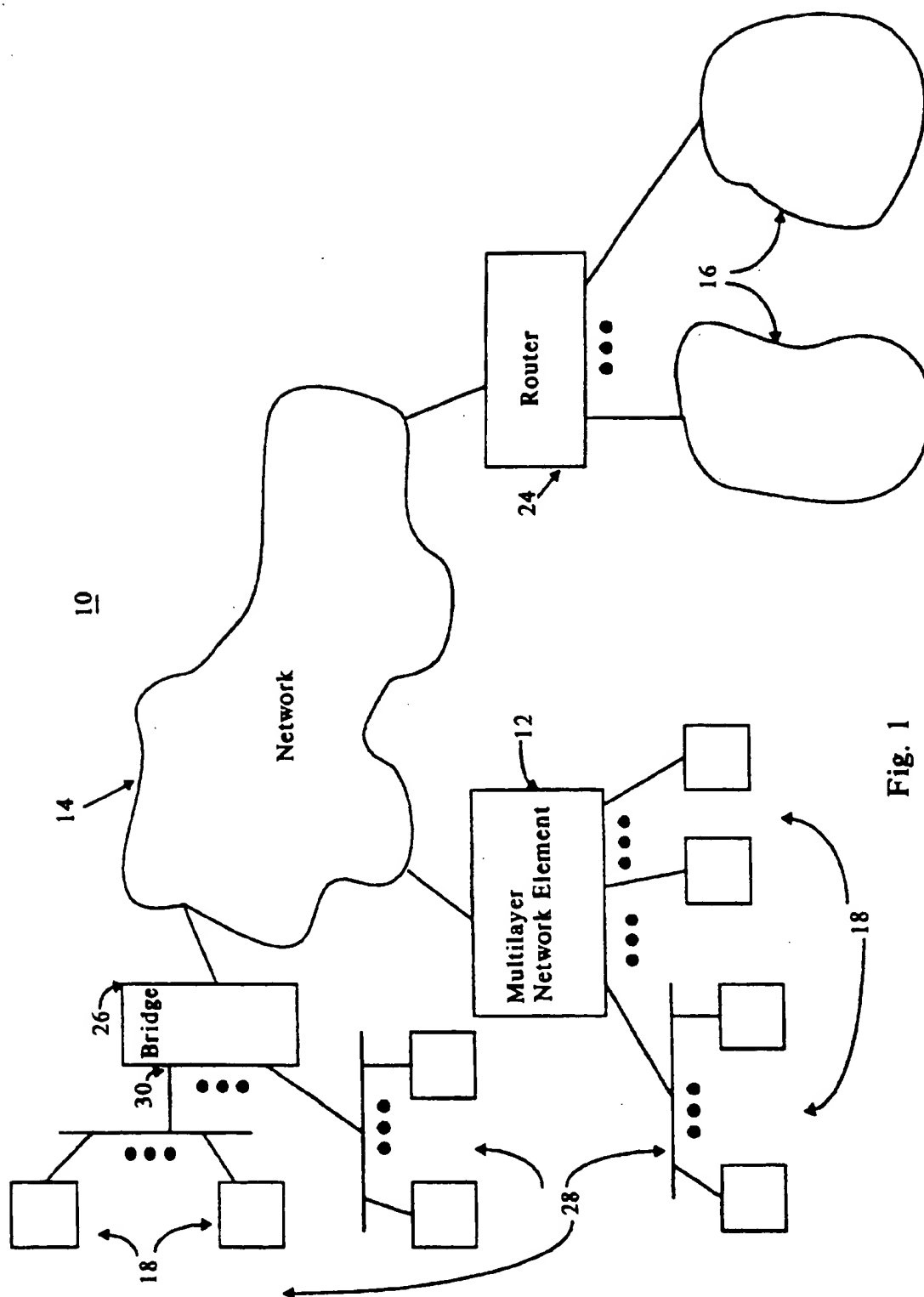


Fig. 1

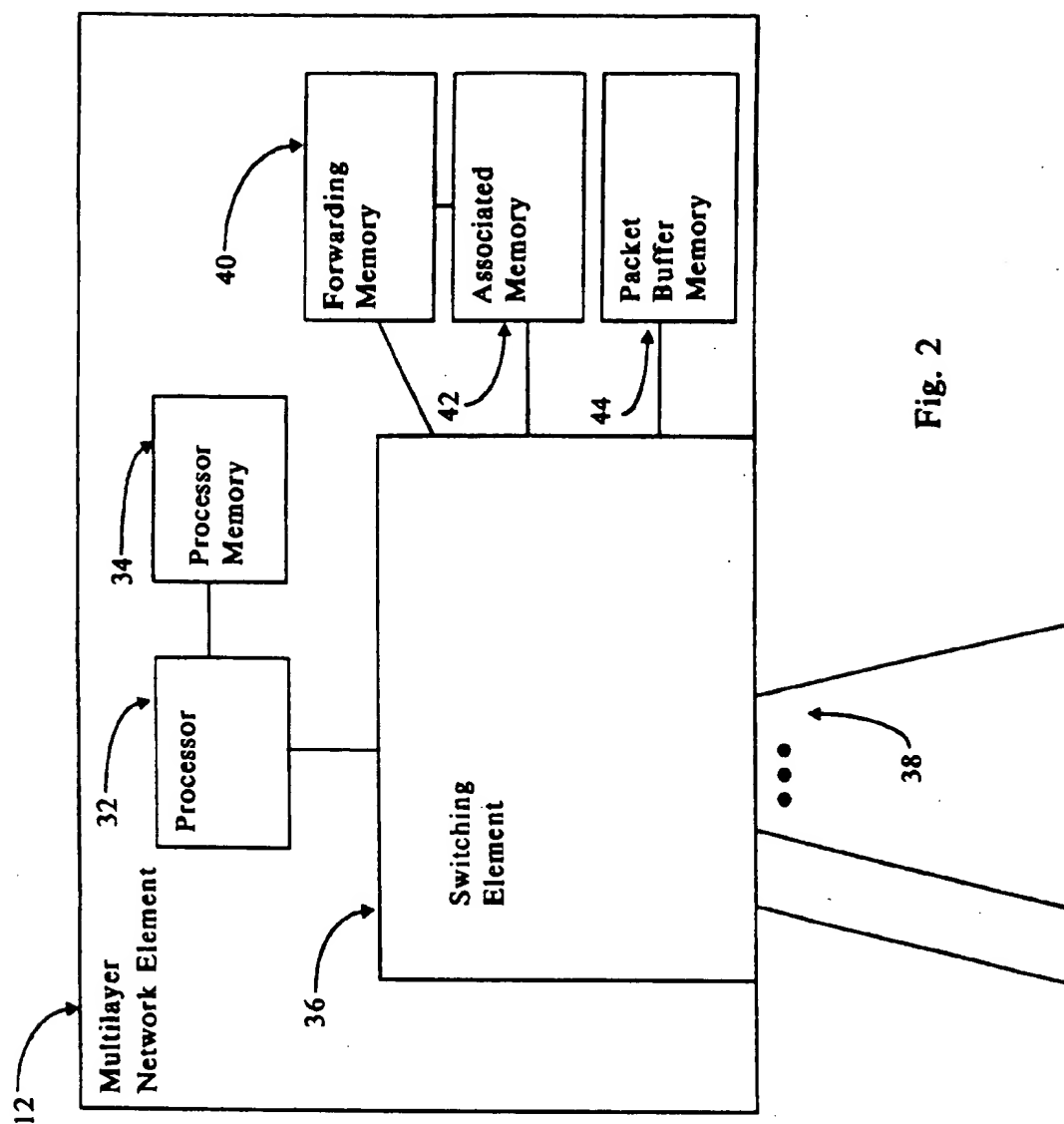


Fig. 2

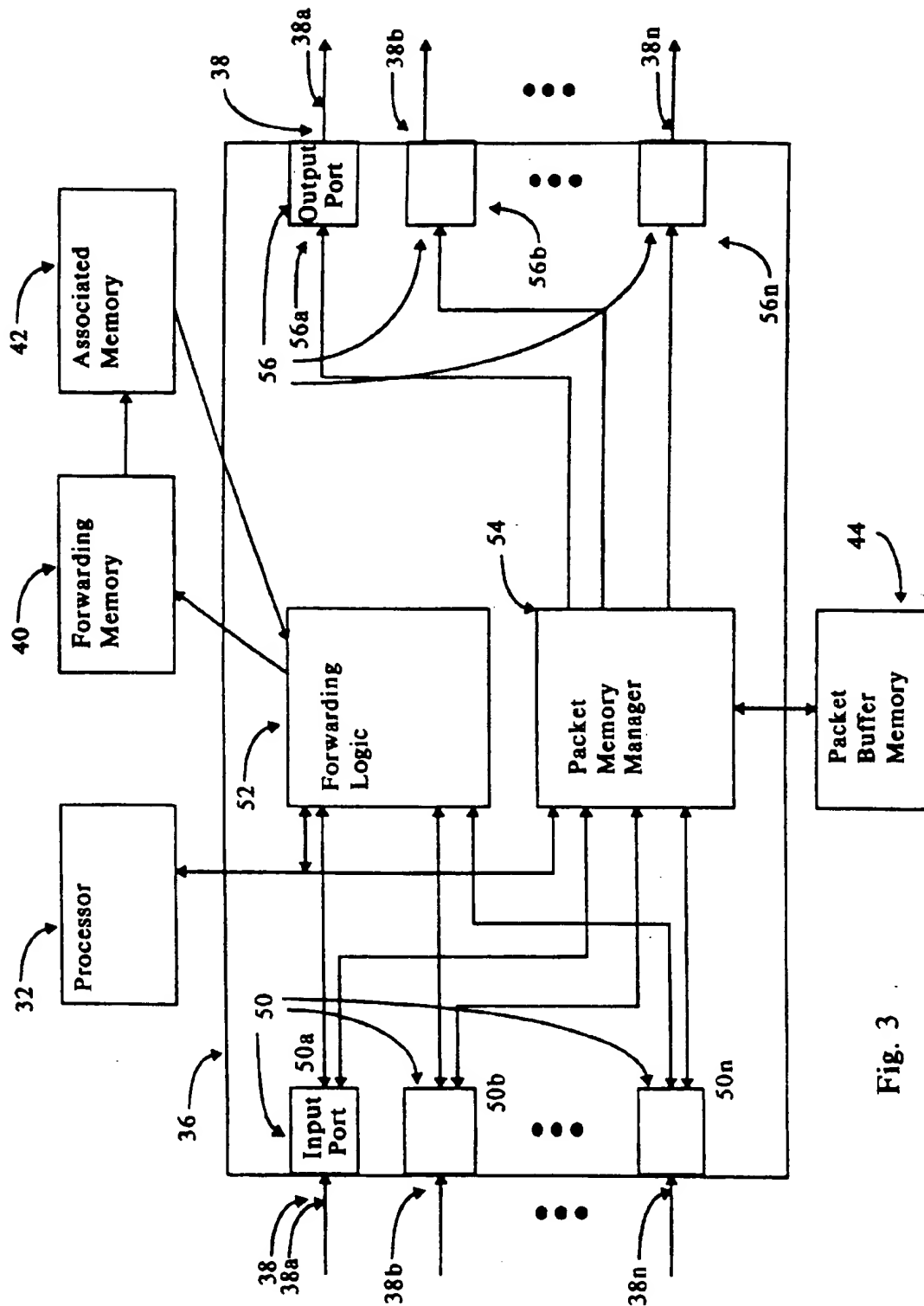


Fig. 3

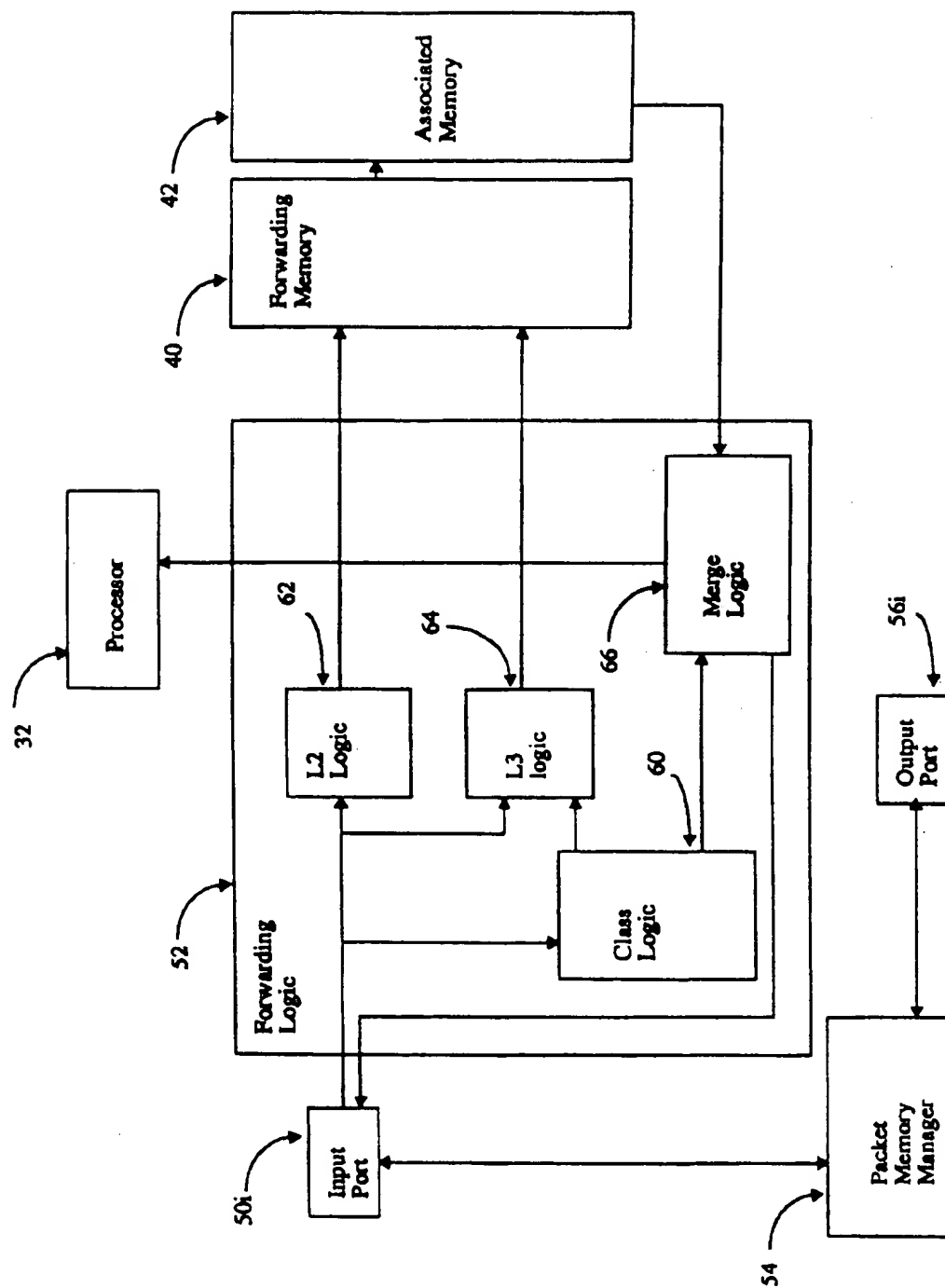


Fig. 4

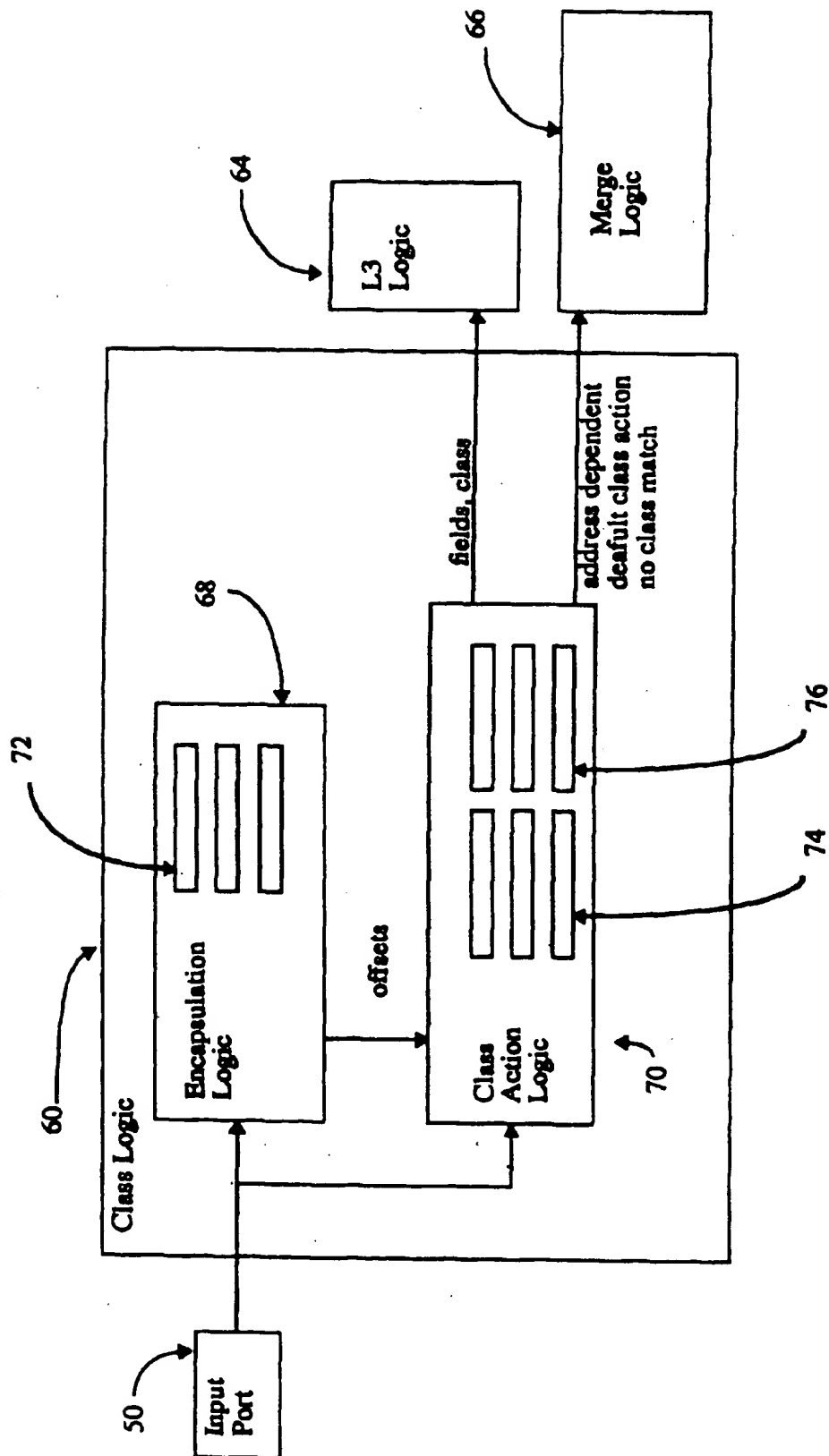


Fig. 5

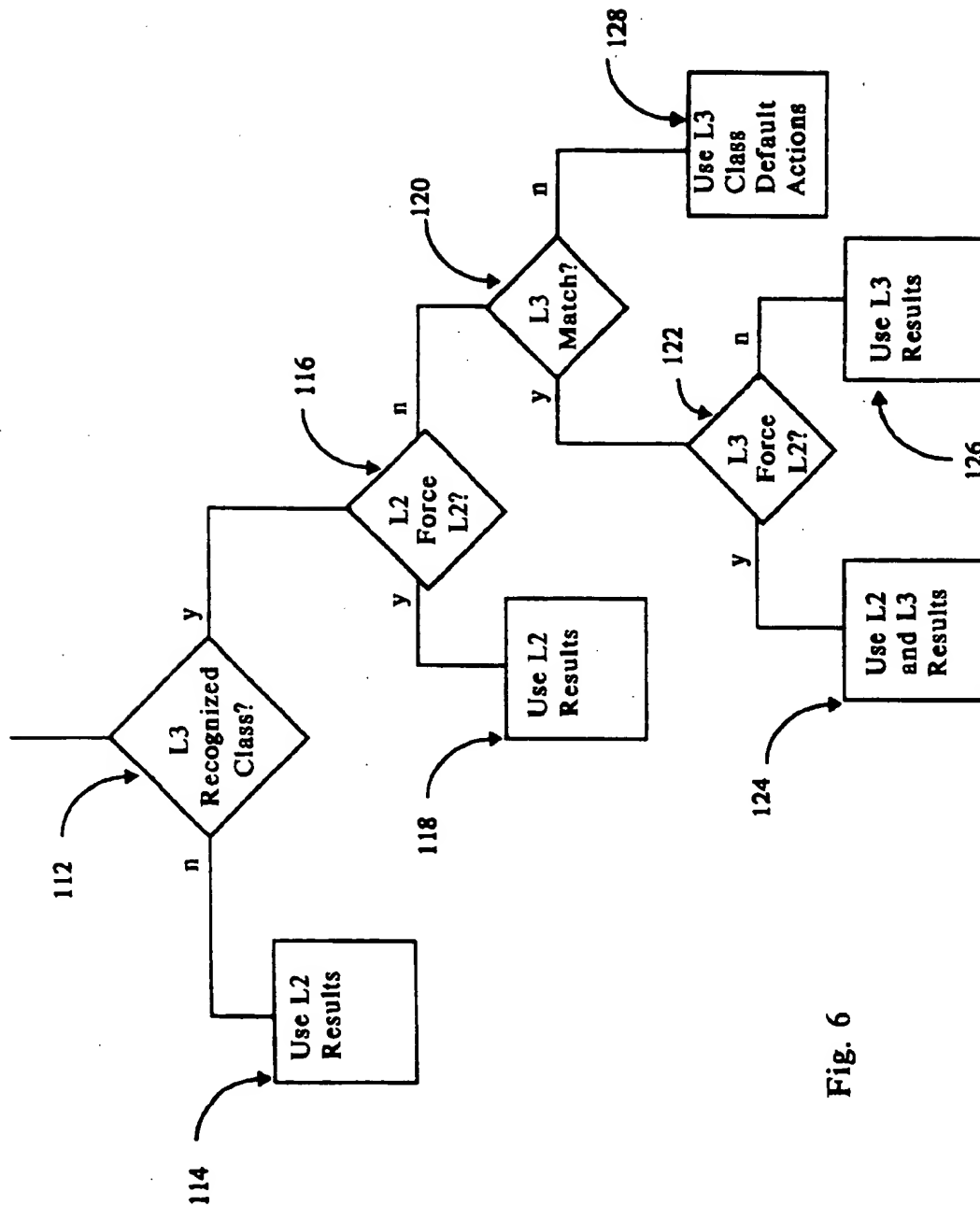


Fig. 6

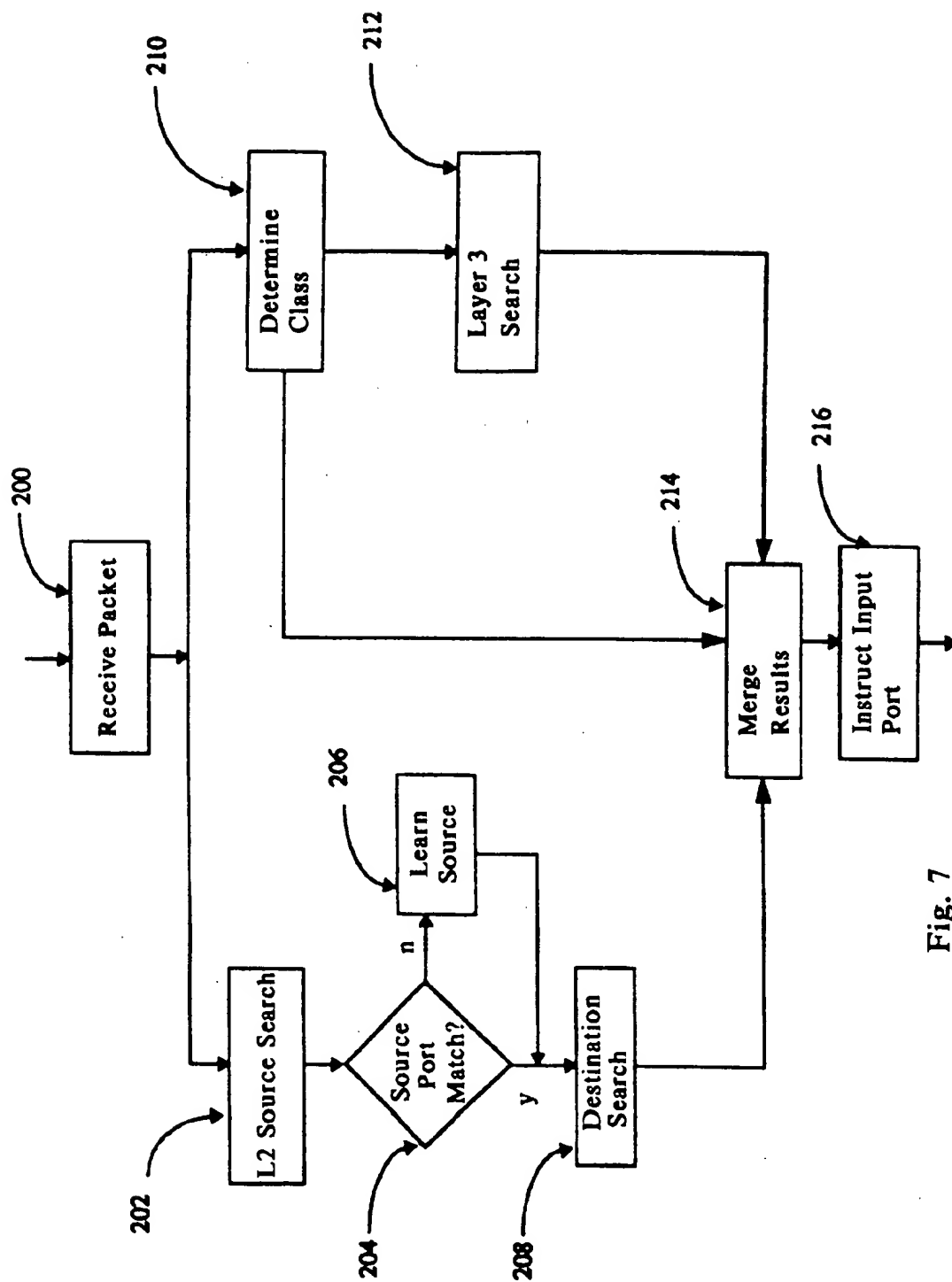


Fig. 7

SYSTEM AND METHOD FOR A MULTI-LAYER NETWORK ELEMENT

FIELD OF THE INVENTION

The present invention relates in general to packet forwarding within a network and, in particular, to a system and method for forwarding packets using multi-layer information.

BACKGROUND OF THE INVENTION

Communication between computers has become an important aspect of everyday life in both private and business environments. Networks provide a medium for this communication and further for communication between various types of elements connected to the network such as servers, personal computers, workstations, memory storage systems, or any other component capable of receiving or transmitting data to or from the network. The elements communicate with each other using defined protocols that define the orderly transmission and receipt of information. In general, the elements view the network as a cloud to which they are attached and for the most part do not need to know the details of the network architecture such as how the network operates or how it is implemented. Ideally, any network architecture should support a wide range of applications and allow a wide range of underlying technologies. The network architecture should also work well for very large networks, be efficient for small networks, and adapt to changing network conditions.

Networks can be generally be differentiated based on their size. At the lower end, a local area network (LAN) describes a network having characteristics including multiple systems attached to a shared medium, high total bandwidth, low delay, low error rates, broadcast capability, limited geography, and a limited number of stations, and are generally not subject to post, telegraph, and telephone regulation. At the upper end, an enterprise network describes connections of wide area networks and LANs connecting diverse business units within a geographically diverse business organization.

To facilitate communication within larger networks, the networks are typically partitioned into subnetworks, each sharing some common characteristic such as geographical location or functional purpose, for example. The partitioning serves two main purposes: to break the whole network down into manageable parts and to logically (or physically) group users of the network. Network addressing schemes may take such partitioning into account and thus an address may contain information about how the network is partitioned and where the address fits into the network hierarchy.

For descriptive and implementive purposes, a network may be described as having multiple layers with end devices attached to it, communicating with each other using peer-to-peer protocols. The well-known Open Systems Interconnection (OSI) Reference Model provides a generalized way to view a network using seven layers and is a convenient reference for mapping the functionality of other models and actual implementations. The distinctions between the layers in any given model is clear, but the implementation of any given model or mapping of layers between different models is not. For example, the standard promulgated by the Institute of Electrical and Electronics Engineers (IEEE) in its 802 protocols defines standards for LANs and its definitions overlap the bottom two layers of the OSI model.

In any such model, a given layer communicates either with the same layer of a peer end station across the network,

or with the same layer of a network element within the network itself. A layer implements a set of functions that are usually logically related and enable the operation of the layer above it.

The relevant layers for describing this invention include OSI Layers 1 through 4. Layer 1, the physical layer, provides functions to send and receive unstructured bit patterns over a physical link. The physical layer concerns itself with such issues as the size and shape of connectors, conversion of bits to electrical signals, and bit-level synchronization. More than one type of physical layer may exist within a network. Two common types of Layer 1 are found within IEEE Standard 802.3 and FDDI (Fiber Distributed Data Interface).

Layer 2, the data link layer, provides support for framing, error detecting, accessing the transport media, and addressing between end stations interconnected at or below layer 2. The data link layer is typically designed to carry packets of information across a single hop, i.e., from one end station to another within the same subnet, or LAN.

Layer 3, the network layer, provides support for such functions as end to end addressing, network topological information, routing, and packet fragmentation. This layer may be configured to send packets along the best "route" from its source to its final destination. An additional feature of this layer is the capability to relay information about network congestion to the source or destination if conditions warrant.

Layer 4, the transport layer, provides application programs such as an electronic mail program with a "port address" which the application can use to interface with the data link layer. A key difference between the transport layer and the lower layers is that an application on a source end station can carry out a conversation with a similar application on a destination end station anywhere in the network; whereas the lower layers carry on conversations with end stations which are its immediate neighbors in the network. Layer 4 protocols also support reliable connection oriented services, an example Layer 4 protocol providing such services is the Transport Control Protocol (TCP).

Different building blocks exist for implementing networks that operate at these layers. End stations are the end points of a network and can function as sources, destinations and network elements or any other intermediate point for forwarding data received from a source to a destination.

At the simplest level are repeaters which are physical layer relays which simply forward bits at Layer 1.

Bridges represent the next level above repeaters and are data link layer entities which forward packets within a single LAN using look-up tables. They do not modify packets, but just forward packets based on a destination. Most bridges are learning bridges. In these bridges, if the bridge has previously learned a source, it already knows to which port to forward the packet. If the bridge has not yet forwarded a packet from the destination, the bridge does not know the port location of the destination, and forwards the packet to all unblocked output ports, excluding the port of arrival. Other than acquiring a knowledge of which ports sources are transmitting packets to, the bridge has no knowledge of the network topology. Many LANs can be implemented using bridges only.

Routers are network layer entities which can forward packets between LANs. They have the potential to use the best path that exists between sources and destinations based on information exchanged with other routers that allow the routers to have knowledge of the topology of the network. Factors contributing to the "best" path might include cost, speed, traffic, and bandwidth, as well as others.

Brouters are routers which can also perform as bridges. For those layer 3 protocols of which the brouter knows, it uses its software to determine how to forward the packet. For all other packets, the brouter acts as a bridge.

Switches are generalized network elements for forwarding packets wherein the composition of the switch and whether it implements layer 2 or layer 3 is not relevant.

Typically, bridges forward packets in a flat network without any cooperation by the end stations, because the LAN contains no topological hierarchy. If a LAN, for example, is designed to support layer 3 functionality, then routers are used to interconnect and forward packets within the LAN.

Bridges cannot use hierarchical routing addresses because they base their forwarding decisions on media access control (MAC) addresses which contain no topological significance. Typically MAC addresses are assigned to a device at its time of manufacture. The number of stations that can be interconnected through bridges is limited because traffic isolation, bandwidth, fault detecting, and management aspects become too difficult or burdensome as the number of end stations increases.

Learning bridges self-configure, allowing them to be "plug and play" entities requiring virtually no human interaction for setup. Routers, however, require intensive configuration, and may even require configuration activities at the end nodes. For example, when a network utilizes the Transmission Control Protocol/Internet Protocol (TCP/IP), each end node must manually receive its address and subnet mask from an operator, and such information must be input to the router.

Generally, as the size and complexity of a network increases, the network requires more functionality at the higher layers. For example, a relatively small LAN can be implemented by using Layer 1 elements such as repeaters or bridges, while a very large network uses up to and including Layer 3 elements such as routers.

A single LAN is typically insufficient to meet the requirements of an organization because of the inherent limitations: (1) on the number of end stations that can be attached to a physical layer segment; (2) the physical layer segment size; and (3) the amount of traffic, which is limited because the bandwidth of the segment must be shared among all the connected end stations. In order to overcome these constraints, other network building blocks are required.

As briefly described above, when the number of end stations in a network increases, the network may be partitioned into subnetworks. A typical address in a partitioned network includes two parts: a first part indicating the subnetwork; and a second part indicating an address within the subnetwork. These types of addresses convey topological information because the first part of the address defines geographical or logical portions of the network and the second part defines an end station within the subnetwork portion. Routing with hierarchical addressing involves two steps: first packets are routed to the destination's subnetwork; and second packets are forwarded to the destination within the subnetwork.

An end station receives a unique data link address—the MAC address—at the time of manufacture, allowing the end station to attach to any LAN within a bridged network without worrying about duplicate addresses. Data link addresses therefore cannot convey any topological information. Bridges, unlike routers, forward packets based on data link addresses and thus cannot interpret hierarchical addresses.

The current Internet is being forced to deal with increasing numbers of users and increasing demands of multimedia applications. Future networks will be required to support even higher bandwidth, larger numbers of users, and traffic classification requirements by the network. Statistical studies show that the network domain as well as the number of workstations connected to the network will grow at a faster rate in future. The trend is also to support multiple traffic types with varied characteristics on a same physical link. This calls for more network bandwidth and efficient usage of resources. To meet the bandwidth requirement, the speed on the networks is on the upward trend, reaching to gigabit speeds.

Network designers frequently use one particular combination of ISO Layer 2 and Layer 3 because of the success of the Internet and the increasing number of products and networks using the Internet. Specifically, in a typical Internet-associated network, designers combine an implementation in accordance with the IEEE 802 Standard (which overlaps ISO Layer 1 and Layer 2) with the Internet Protocol (IP) network layer. This combination is also becoming popular within enterprise networks such as intranets.

Supporting this combination by building networks out of layer 2 network elements provides fast packet forwarding but has little flexibility in terms of traffic isolation, redundant topologies, and end-to-end policies for queuing and administration (access control). Building such networks out of layer 3 elements alone sacrifices performance and is impractical from the hierarchical point of view because of the overhead associated with having to parse the layer 3 header and modify the packet if necessary. Furthermore, using solely layer 3 elements forces an addressing model with one end station per subnet, and no layer 2 connectivity between the end stations.

Networks built out of a combination of layer 2 and layer 3 devices are used today, but suffer from performance and flexibility shortcomings. Specifically, with increasing variation in traffic distribution (the role of the "server" has multiplied with browser-based applications), the need to traverse routers at high speed is crucial.

The choice between bridges and routers typically results in significant tradeoffs (in functionality when using bridges, and in speed when using routers). Furthermore, the service characteristics, such as priority, within a network are generally no longer homogeneous, despite whether traffic patterns involve routers. In these networks, differing traffic types exist and require different service characteristics such as bandwidth, delay, and etc.

To meet the traffic requirements of applications, the bridging devices should operate at line speeds, i.e., they operate at or faster than the speed at which packets arrive at the device, but they also must be able to forward packets across domains/subnetworks. Even through current hybrid bridge/router designs are able to achieve correct network delivery functions, they are not able to meet today's increasing speed requirements.

What is needed is a switch or network element that forwards both layer 2 and layer 3 packets quickly and efficiently both within a subnetwork and across subnetworks, and to other networks. Further, a network element is needed that can forward layer 3 packets at wire-speed, i.e., as fast as packets enter the network element. Additionally, a network element is needed that allows layer 2 forwarding within a subnetwork to have the additional features available in layer 3 routing and to provide certain quality of service for applications within the subnetwork, such as priority and bandwidth reservation.

SUMMARY OF THE INVENTION

The present invention enables the above problems to be substantially overcome by providing a system and method for an multi-layer network element for forwarding received packets to one or more appropriate output ports.

An embodiment of the present invention includes a method of forwarding a packet entering from an input port to one or more appropriate output ports based on single searches of an associative memory for each layer. The associative memory contains certain quality of service information that may be applied to any layer.

A packet is received on an input port, and from the packet a first search key is created based on the header of the packet. An associative memory lookup is performed for the first search key, which results in two potential forwarding decisions for the packet. If the first search key matches an entry to a destination address found in the first search key, i.e., a matching entry is found in the associative memory, then the potential output port or ports are those associated with the destination address as found in the associative memory. If the destination address does not match any entry in the associative memory, then all ports except the incoming port are candidates for the potential output port or ports.

The packet is also categorized by class to aid in creating a second search key. Packets of a particular class share common characteristics, such as what portions of the header will be used to create the second search key. A class also defines certain default forwarding information for packets within the class. The default information may include certain quality of service information.

An associative memory lookup is performed using the second search key. The results of this second search, the first search, and the default information are combined to determine which of the potential output port or ports as proffered by the three searches is the most appropriate for this packet. The packet is then forwarded to the appropriate output port or ports.

In some instances, the second search key yields no match in the associative memory. In these cases, the default information is combined with the results of the first search. Furthermore, the results of the first search may override any of the other forwarding information; and the results of the second search may force the results of the first search to be used to forward the packet.

In one embodiment, the invention implements forwarding of layer 2 and layer 3 packets. In this embodiment, the first search key includes information about layer 2 destination addresses and the second search key and default information include information about layer 3 and possibly layer 4.

Such an implementation, in one embodiment, allows certain quality of service to be applied to layer 2 forwarding in the following manner. When a packet enters the network element as a layer 2 packet, the first search key will result in layer 2 forwarding information being output from the associative memory. The class of the packet will be determined and the packet provided with default class information that may include certain quality of service information. The second search key, however, may not yield any results from the associative memory because an entry in the memory has not yet been created by the central processing unit. In this instance, merge logic will use the layer 2 forwarding result but also use the quality of service information from the default forwarding information. Such a feature allows the network element to be configured to provide quality of service to layer 2 traffic within a subnetwork.

Still other embodiments of the present invention will become readily apparent to those skilled in the art from the following detailed description, wherein is shown and described only the embodiments of the invention by way of illustration of the best modes contemplated for carrying out the invention. As will be realized, the invention is capable of other and different embodiments and several of its details are capable of modification in various obvious respects, all without departing the spirit and scope of the present invention. Accordingly, the drawings and detailed description are to be regarded as illustrative in nature and not as restrictive.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a system incorporating a multi-layer network element according to the invention.

FIG. 2 illustrates the multi-layer networking element of FIG. 1.

FIG. 3 illustrates the switching element of the multi-layer network element in more detail.

FIG. 4 illustrates the forwarding logic of the switching element in more detail.

FIG. 5 illustrates the class logic of FIG. 4 in more detail.

FIG. 6 illustrates the process used in determining which information dictates a packet's path through the multi-layer network element.

FIG. 7 illustrates the information dependency in determining how to forward a packet out of the network element.

DETAILED DESCRIPTION

FIG. 1 illustrates a system incorporating a multi-layer network element according to the present invention. The system includes the multi-layer network element, various networks, end stations, routers, and bridges. By way of example and as broadly embodied and described herein, a system 10 incorporating a multi-layer network element 12 according to the present invention includes networks 14 and 16, end stations 18, router 24, bridge 26, and local area networks (LAN) 28.

The bridge 26 connects some of the LANs 28 and end stations 18 to the network 14 and to each other. The bridge 26 may be a conventional learning bridge. The bridge 26 keeps track of the addresses of the end stations 18 that transmit a packet showing up on one of ports 30 to the bridge 26. The end stations 18 may be any device capable of sending or receiving packets of information. Typically, the end stations 18 are personal computers, workstations, printers, servers, and/or any other device that can be connected to a network.

The bridge 26 initially does not know on which of its ports packet destinations are located, and must flood an incoming packet to all ports in order to properly forward the packet. Once the bridge 26 receives a packet destined for an address it already recognizes, the bridge 26 knows what port the destination is on so that it does not have to flood the packet on all outgoing ports. Eventually, the bridge 26 has learned enough addresses to all but eliminate the amount of flooding needed on the ports. Of course, any time an end station 18 changes ports on the bridge 26, the bridge 26 must relearn the end station 18's port.

The bridge 26 typically does not modify the packet, contains no information about the topology of the network 14, and examines few parts of the packet header. The bridge 26 operates quickly because it makes no modifications to the packet and is only concerned with learning sources and forwarding to destinations. Typically, bridges 26 use look-up tables to search for sources and destinations.

The router 24 connects the network 14 to the networks 16. Only one router 24 is illustrated by way of example, but there may be many routers connecting other networks or end stations 18. The router 24 provides the communication necessary between the network 14 and the networks 16 and may be a conventional router. Such routers include layer 3 functionality for forwarding packets to an appropriate destination including route calculation, packet fragmentation, and congestion control. Routers of this type are described, for example, in *Interconnections: Bridges and Routers* by Radia Perlman published by Addison-Wesley. The router 24 must have knowledge of the topology of the network in order to determine the best route for packets. The router 24's knowledge of the network is gained through topological information passed between multiple such routers 24 connected to the network 14.

Software running on the router 24 parses an incoming packet to determine various characteristics about the packet, including the type of the protocol being used and the source and destination(s). Other determinations based on examining the packet may be necessary, such as priority and quality of service (QoS) factors such as priority and bandwidth reservation. The router 24 then uses the extracted information and computes the next destination for the packet based on topology and route information that is stored in the memory of the router 24. The router 24 also applies QoS rules and actions.

The router 24's process for calculating the next destination may require many accesses to memory and computation of the route from that information. Furthermore, the packet is typically received and stored while any processing is taking place. After the router 24 has determined what actions are necessary on the packet, any modifications are made to the packet as stored in the memory or on the way out of the router 24. The routers 24 are typically required to replace the layer 2 source and destination of the packet, update any checksums of the packet, and handle any issues related to packet lifetime.

To carry out the functions that the conventional router 24 performs, the software examines memory locations, make modifications to the packet, and calculate new values for some fields. Such actions provide increased functionality beyond simple packet forwarding like that found in bridges 26 such as determining the best route for the packet, providing QoS features; however, in conventional routers 24 such actions take up valuable time.

The network 14 provides communication paths for all of the elements connected to it. In the example of FIG. 1, the elements include the multi-layer network element 12, router 24, and bridge 26. Any number of elements could be connected to the network 14 in a multitude of ways. FIG. 1 illustrates only one possible combination. The elements connected to the network 14 do not require the network 14 to be of any particular size or configuration. For the end stations 18 and the bridge 26, a detailed topological knowledge of the network 14 is not required.

The multi-layer network element 12 according to the present invention connects various elements to the network 14 and to each other. As illustrated by way of example, the multi-layer network element 12 connects a LAN 28, the end stations 18, and the network 14. The multi-layer network element 12 combines the functions of both a bridge and a router. Functioning as a router, the multi-layer network element 12 contains topological information about network 14 to intelligently route a packet to its destination while providing associated layer 3 functionality typically found in

a router 24. Functioning as a bridge, the multi-layer network element 12 learns source/port combinations to forward layer 2 packets. The multi-layer network element 12 differs from conventional bridge/router combinations in that certain layer 3 processing operates as quickly as layer 2 switching found in the bridge 26.

FIG. 2 illustrates the multi-layer network element 12 of FIG. 1 in more detail. The multi-layer network element 12 according to one embodiment of the invention includes a processor 32, a processor memory 34, a switching element 36, a plurality of network element ports 38, a forwarding memory 40, an associated memory 42, and packet buffer memory 44. The end stations 18, the LAN 28, and the network 14 are connected to the multi-layer network element 12 using a plurality of network element ports 38. Other multi-layer network elements 12 may also be connected to the multi-layer network element 12.

The switching element 36 is also connected to the processor 32, the forwarding memory 40, the associated memory 42, and the packet buffer memory 44. The processor 32 is also connected to the memory 34. Forwarding memory 40 and associated memory 42 is connected to each other as well to as switching element 36.

The switching element 36 performs most of the packet forwarding functions using both layer 2 and layer 3 information, and possibly also some layer 4 information, stored in forwarding memory 40 and associated memory 42, without having to rely on the processor 32 to calculate routes or determine appropriate actions on every packet.

The processor 32 performs tasks that the switching element 36 is not equipped to handle. For example, when new layer 3 routes must be calculated, the processor 32 uses processor memory 34, which contains detailed information about the topology of any networks reachable from the multi-layer network element 12. The processor 32 makes its computations primarily using software programming units in conjunction with accesses to the memory 34. The switching element 36 makes its decisions primarily in hardware, using the forwarding memory 40 and the associated memory 42. The forwarding memory 40 and the associated memory 42 contain only a portion of the information contained in the memory 34, and are configured for quick access and retrieval.

FIG. 3 illustrates a detailed view of the switching element 36 and its connections to the processor 32, the plurality of network element ports 38a-n, the forwarding memory 40, the associated memory 42, and the packet buffer memory 44. The switch element 36 includes input ports 50a-n, a forwarding logic 52, a packet memory manager 54, and output ports 56a-n. Each input port 50i and output port i corresponds to a network element port 38i. Each of the inputs ports 50 also connects to both the forwarding logic 52 and the packet memory manager 54.

For a given i, an input port 50i receives packets from its respective multi-layer network element port 38i and tests the packets for correctness. If the packet is ill formed, it is discarded. Packets passing this initial screening are temporarily buffered by the input port 50i. Once the input port 50i has buffered at least the first 64 bytes of the received packet, the input port 50i passes the header to the forwarding logic 52.

The forwarding logic 52 is connected to the processor 32, the forwarding memory 40, and the associated memory 42. The forwarding logic 52 performs several functions. It initially screens the packet to determine whether the packet is encapsulated, by for example Subnetwork Access Proto-

col (SNAP), or whether the packet is tagged, for example, by a virtual LAN (VLAN) identifier. If the packet is either of those two types, the forwarding logic 52 uses offset information to locate appropriate layer header information needed for further processing.

The forwarding logic 52 also searches the forwarding memory 40 for matches at layer 2 and/or layer 3. The search may also include some information at layer 4. In the preferred embodiment, the forwarding memory 40 is a content-addressable memory (CAM) storing information about both layer 2 and layer 3 switching, and may contain some layer 4 information. If a match is found, data stored in associated memory 42 and pointed to by the matching entry in the forwarding memory 40 serves to define the actions that the switching element 36 must do to forward the packet to the appropriate destination(s).

In another embodiment, the forwarding memory 40 could be implemented using an sequentially address random access memory. In this embodiment, a hashing function would be performed on the particular key. The resulting hashed value would be an address into the memory 42 associated with the pre-hashed key.

In still another embodiment, the forwarding memory 40 and the associated memory 42 could be contained in a single random access memory. In one implementation of that single random access memory, the entries could be accessed sequentially, requiring a hash-front end. Another implementation of that single random access memory could be a CAM.

The packet memory manager 54 is connected to the packet buffer memory 44, the input port 50i, and the output port 56i. As indicated above, each output port 56i corresponds to one of the plurality of multi-layer network element ports 38i. While illustrated as separate units, the input port 50i and output port 56i corresponding to a particular multi-layer network element port 38i are tightly coupled since information flows both ways through the network element ports 38.

After the forwarding logic 52 has determined what to do with the packet, it passes that information to the input port 50i. If the input port 50i does not filter the packet, then it requests pointer to free memory locations in the packet buffer memory 44 from the packet memory manager 54. The packet memory manager 54 responds by providing location addresses of free memory space in the packet buffer memory 44. The input port 50i then requests a write access from the packet memory manager 54 and sends the pointer and the data to the packet memory manager 54.

In some instances, the input port 50i must make modifications to the packet as instructed to do so from the forwarding logic 52. The input port 50i makes these modifications prior to the packet being stored in the packet buffer memory 44. When requested by the input port 50i, the packet memory manager 54 places the packet into the appropriate address location specified by the input port 50i. The input port 50i then passes information about where the packet is stored to the appropriate output ports 56 as determined from the information received at the input port 50i from the forwarding logic 52.

In a preferred embodiment, the appropriate output ports may include no output ports or one or more output ports. The output port 56i requests and receives packets from the packet manager 54, and transmits the packet to its associated network element port 38i when the conditions for transmission are met. In some instances, the output port 56i must place its MAC address as the source address on outgoing

packets. If this situation is dictated by the results from the forwarding logic 52 as passed to the input port 50i, the input port 50i places such an indication in the packet buffer memory 44. The output port 56i detects this indication and replaces the address as the packet leaves the output port 56i. Thus, only minor modifications to the packets are necessary on the output side of the switching element 36.

According to the above embodiment, when the forwarding memory 40 contains matching entries for layer 2 switching or layer 3 routing, the multi-layer network element 12 will operate at wire-speed. Wire-speed is defined by the speed at the maximum packet rate at which a given layer 1 and layer 2 combination can transport packets. If an element connected to a network can process packets as fast as they enter the element or faster, then the element operates at wire speed.

In a preferred embodiment, the network element 12 processes packets for a worst-case scenario of a steady stream of 64-byte packets entering all input ports 50 simultaneously. If the layer 3 information is not contained in the forwarding memory 40, the packet is forwarded using layer 2 information and then processed according to conventional layer 3 processing by software in the processor 32.

Unlike conventional layer 3 processing, the processor 32 may update the forwarding memory 40 by placing new layer 3 entries as they are learned and created. Any packets matching the new entries are forwarded at wire-speed, i.e. forwarding decisions are made for a packet before the next packet arrives.

While the discussion of this invention is described using layer 2 and a combination of layers 3 and 4, one skilled in the art would recognize that searching on and creating entries in the forwarding memory 40 for any portion of a packet or its header, or any combination thereof, readily flows from the description. Thus, this invention is not limited to any specific implementation of layers according to the ISO standard.

FIG. 4 illustrates the forwarding logic 52 in more detail. The forwarding logic 52 includes class logic 60, layer 2 (L2) logic 62, layer 3 (L3) logic 64, and merge logic 66. The input port 50i connects to the class logic 60, the L2 logic 62, the L3 logic 64, and the merge logic 66. Only one input port 50i is shown for simplification, though all input ports 50 are connected in a similar manner. Preferably, the forwarding logic 52 is not duplicated for each input port 50i. Instead, all input ports 50 arbitrate for access to the forwarding logic 52.

The L2 logic 62 is connected to the forwarding memory 40 and is responsible for creating a key to match against the entries stored in the forwarding memory 40 for layer 2 forwarding decisions. Depending on the configuration of the forwarding memory 40, the key may be applied against all or some of the entries of the forwarding memory 40.

During operation, the input port 50i receives a packet from the multi-layer network element port 38i and sends the header plus the input port 50i identifier to the forwarding logic 52. The forwarding logic 52 first searches the forwarding memory 40 to determine whether the forwarding memory 40 contains an entry for the layer 2 source transmitting the packet. A matching entry will exist if the multi-layer network element 12 has previously received a packet from the same layer 2 source and has learned which port it is connected to. If no matching entry exists, the forwarding logic 52 performs a learn function by placing an entry in the forwarding memory 40 including the source address. The forwarding logic 52 signals the processor 32 that it has learned a new source address. In some instances,

the layer 2 source will exist in the forwarding memory 40, but will be associated with a different input port 50i than the input port 50i of the incoming packet. In this instance, no matching entry will exist in the forwarding memory 40 because a match depends on both the layer 2 source and the input port 50i.

The forwarding logic 52 also searches the forwarding memory 40 for an entry indicating the port of the destination address. If no match is found, then the forwarding logic 52 instructs the input port 50i to flood the packet to all of the active output ports 56.

For the layer 2 information described above in the preferred embodiment, the forwarding memory 40 contains the values of the MAC addresses of the sources and a pointer to a corresponding entry in the associated memory 42. The forwarding memory 40 may also contain additional layer 2 information such as a VLAN identifier if tagged packets are being used. The associated memory 42 contains more information about its corresponding entry in the forwarding memory 40. Layer 2 information in the forwarding memory 40 is preferably limited to the least amount of information necessary to make a layer 2 search. In a layer 2 search, this information is preferably just the MAC address and the input port 50i, but the CAM may also contain any information relating to tagged addressing.

In a preferred embodiment, the forwarding memory 40 allows multiple matches for a layer 2 search. The processor 32 ensures that the order of the entries is such that if an address/port combination exists in the forwarding memory, that entry is selected. If the particular source/port combination is not found, then a match may occur including VLAN information so that any layer 2 destination search will at least match a known VLAN or an unknown VLAN entry, each of which define the output ports 56 for flooding in its respective entry.

The L3 logic 64 is connected to the forwarding memory 40 and is responsible for creating a key to match against the entries stored in the forwarding memory 40 for layer 3 forwarding decisions. As with the L2 search key, the L3 key may be applied against all or some of the entries of the forwarding memory 40.

To create the key, the L3 logic 64 uses information from the input port 50i including the packet header and an input port 50i identifier, and information from the class logic 60. The merge logic 66 is connected to the class logic 60, the associated memory 42, the packet memory manager 54, and the processor 32. The merge logic 66 uses information from the class logic 60 and information output from the associated memory 42 to instruct the input port 50i what to do to properly forward the packet to its appropriate destination(s). In some instances, there is no appropriate destination and the packet is discarded. In other instances, the merge logic 66 will signal the processor 32 that it must perform some task in response to the received packet.

Layer 3 switching, while more complex, is similar to layer 2 switching. The forwarding logic 52 searches the forwarding memory 40 for a matching entry to a layer 3 search key created by the L3 logic 64. If a match exists, the information in the associated memory 42 is used by the merge logic 66 to instruct the input port 50i what to do with the packet. If the search provides no match, the switching element 36 forwards the packet as a bridge and may pass all or portions of the packet to the processor 32 for further processing. The L3 logic 64 creates the search key using information from the packet header, the input port 50i, and the class logic 60.

The class logic 60 examines information in the packet header to determine any encapsulation information and to

determine a class for the layer 3 information and is illustrated in more detail in FIG. 5. The class logic 60 includes the encapsulation logic 68 and the class action logic 70. Each input port 50i is connected to both the encapsulation logic 68 and the class action logic 70. The class action logic 70 is connected to the encapsulation logic 68, the L3 logic 64, and the merge logic 66.

The encapsulation logic 68 is responsible for examining the packet header and determining any offsets into the header for the layer 3 and layer 4 information, if needed. The encapsulation logic 68 includes class filters 72 to determine any offsets into the packet to identify locations of relevant information. In a preferred embodiment one filter 72 recognizes an implementation in accordance with the IEEE 802.3 Standard Ethernet header, and another filter 72 recognizes an implementation in accordance with the IEEE Standard 802.1q Tagged Ethernet Header, and still another recognizes an LCC SNAP encapsulation. Other encapsulations would become readily apparent to one skilled in the art and could be implemented with additional encapsulation filters 72. The encapsulation logic 68 passes encapsulation offsets to the class action logic 70 so that the class action logic 70 knows from where in the packet to draw the appropriate field information.

The class action logic 70 determines to which class a packet belongs. A class is used by both the L2 and L3 logics to aid in searching and to add to the functionality of the multi-layer network element 12. The L2 logic 62 applies a single class to all layer 2 searches. Layer 3, on the other hand, has a plurality of programmable classes.

The classes help to define a class type and for each class which bytes from the packet header that should be used in creating the layer 3 search key by the L3 logic 64, its priority, and a default class result that defines what should happen if no layer 3 match occurs in the forwarding memory 40.

In a preferred embodiment, there are four possible outcomes when no match occurs. First, the header may be sent to the processor 32. This is contemplated when the possibility of identifying a layer 3 flow exists. Second, the entire packet could be copied to the processor 32. This is contemplated when initially setting a unicast route or to provide firewall protection by initially examining certain routes or flows or when it is unknown where in the packet required information may exist to create search keys. Third, use layer 2 results for forwarding. Fourth, discard the packet. Other action may be possible depending on the configuration of the network or the particular protocol in use as would become readily apparent to one skilled in the art.

Some of the criteria that the classes take into account may be whether the class is considered address dependent or address-independent. Adding a class identifier allows the switching element 36 to respond to varying network situations and greatly simplifies organizing and storing information in the forwarding memory 40.

Representative examples of address independent classes that could be identified by the class logic 60 include: Address Resolution Protocol (ARP); Internet Group Management Protocol (IGMP); Reverse ARP (RARP); Group Address Registration Protocol (GARP); Protocol Independent Protocol (PIM); and Reservation Protocol (RSVP). Representative examples of address dependent classes include: TCP flow; non fragmented UDP flow; fragmented UDP flow; hardware routable IP; and IP version 6. Of course, other protocols could be similarly recognized.

The class logic 60 produces an unambiguous class result for every incoming packet. For an unrecognized protocol,

the class logic 60 will still produce a class result, but that class result signifies an unrecognized protocol and what actions should take place on a packet of this type of class.

Generally, layer 3 flows are address dependent and will contain information beyond just a simple class of traffic. In those instances where additional information has been placed by the processor 32 into the forwarding memory 40, there may be more than one entry for a particular class in the forwarding memory 40. The processor 32 ensures that of the entries matched, the one used is the most appropriate one. Different classes may have different criteria for what is the most appropriate match depending on the type of packets embodied within a particular class. The flexibility allowed by having multiple matching entries in the forwarding memory 40 is further enhanced by ensuring that the best match is provided for a particular flow and because of this feature, different actions will be possible for packets within the same type of class.

In the preferred embodiment, the processor 32 reorders the layer 3 entries when it places any new layer 3 so that the best match for a particular search criteria occurs earliest in the memory. Those skilled in the art will recognize many different implementations to achieve the same result. In one preferred embodiment, the processor 32 ensures that the entry with the longest potential matching key within a particular class is at the top, or earliest, location in the memory. However, the processor 32 may also place an entry above the longest matching entry so that for a particular traffic pattern the most important match may be one that matches many keys. For example, an entry that matches, for a particular class, based on an application port such as "http" and no other information, will take precedence over entries that might match more than just the layer 4 application. Another example might be forcing a match on a particular source within a class type. This might occur when the operator might want to provide packets coming from a particular server with a high priority regardless of the destination or layer 4 application.

In a preferred embodiment, the merge logic 66 directs the input port 50i to take one of the following actions on a packet: filter the packet; forward the packet at layer 2; forward the packet as a layer 3 flow; process the packet as a layer 3 route; and forward the packet as a multicast route. Packets that the merge logic 66 instructs the input port 50i to filter are those that include certain header information determined to be unsupported. Examples of classes whose packets would be forwarded at layer 2 would include a fragmented UDP flow and a class indicating that the header information is unknown. A fragmented UDP operates using layer 2 information because after the first packet, the fragmented packets do not include all relevant information from the layer 4 header information, UDP ports for example. Layer 2 forwarding would be optional for address independent classes depending on the particular class.

The merge logic 66 instructs the input port 50i to use layer 3 flow information for TCP or non-fragmented UDP flows. Flows are those packets forwarded within the subnet to which the multi-layer network element 12 is attached and require no header modification on forwarding. Routes, on the other hand, are packets coming from sources outside the subnet or destined to addresses beyond the subnet such that the header information must be modified prior to forwarding by the multi-layer network element 12. In a preferred embodiment, instructions to forward the packet as a layer 3 route come from the merge logic 66 when the class indicates that the packet is of a class hardware routable IP. In other words, the destination of the incoming packet is recognized

by the class logic 60 of the multi-layer network element 12, and the multi-layer network element 12, must then forward the packet to the next hop destination, which is determined by routing protocols. Those skilled in the art can easily recognize from the invention other situations where such a type of result would be desired.

One feature of the invention is the ability to bridge flows, that is, use the forwarding memory to quickly forward layer 2 packets using layer 3 functionality through the network element 12. Certain flows are particularly suited for this type of activity and include static flows, self-detecting flows, and flows set up by reservation protocols, such as RSVP. Static flows are those set up in advance by the network element 12 operator and define layer 3 functionality for selected layer 2 network traffic and are not subject to aging. Self-detecting flows are a function of the type of application.

Initially, these flows are bridged with no layer 3 functionality because no layer 3 entry matches. The packet header is sent to the processor 32 for examination. The processor 32 analyzes the packet and based on programmed heuristics determines whether and how to create a layer 3 entry in the forwarding memory 42 for the packet type. For example, a "ping" packet would not warrant a layer 3 flow entry because it is, at best, a transient packet.

Protocols like RSVP work to reserve certain service features of the network and signal that a number of packets will follow this same path. In this case, it serves the application using the reservation protocol to forward at layer 2, but add layer 3, or more, functionality like priority to ensure the required class of service through the multi-layer network element 12.

FIG. 6 illustrates preferred results produced by the merge logic 66 using information from the class logic 60 and the associated memory 42. Three results are presently preferred: (1) use the layer 2 forwarding results; (2) use the layer 3 forwarding results; and (3) use the layer 3 results while using the layer 2 topology. In some instances, there may be an identified class, but no matching entry in the forwarding memory 40, in this instance, the default actions for the class are used. Note that the use of layer 3 default results can be considered a subset of using layer 3 forwarding results.

Default results may be set for packets of a class type to provide protection such as that provided by firewall technology. In a firewall application, the multi-layer network element 12 would be programmed to direct any packet of a defined class to the processor 32 for subsequent processing.

Referring to FIG. 6, if the class logic 60 determines that the packet is of an unrecognized class (step 112), then the packet is acted on using the layer 2 results (step 114). If the packet's class is recognized (step 112) and the associated memory 42 or class logic 60 indicates that a layer 2 result should be forced (step 116), then the layer 2 results are used (step 118) regardless of any other information.

If no layer 2 results are forced as a result of the layer 2 search (step 116) and there is a match of the layer 3 key (step 120), then the layer 3 information is checked to determine whether the layer 3 information forces a layer 2 port decision (step 122).

If the layer 3 information forces a layer 2 forwarding result, then the output port is determined by the results of the layer 2 search, however, any other information found in the results of the layer 3 search are applied (step 124) such as QoS factors. If the layer 3 results do not call for forcing a layer 2 forwarding result, then the layer 3 results are passed on to the input port 50i (step 126). If there is no layer 3 match in step 120, then the default actions for the class

generated by the class logic 66 are passed to the input port 50i (step 128). It is also contemplated that a packet is sent to the processor 32 without being forwarded to any output port 56 by the input port 50i when using L3 class default action.

Thus, if the class is recognized and the layer 3 search matches an entry, then the actions defined by the layer 3 search govern the instructions to the input port 50i, even though that might mean that the layer 2 output port results are used. If not, the packet is treated using layer 2 results and the packet or the packet's header might be sent to the processor 32 for subsequent processing of the layer 3 information, if desired.

If the information coming out of associated memory 42 for a layer 3 match indicates a force layer 2 result, then packet forwarding is done using the layer 2 results, but any information relating to quality of service may still be implemented on a layer 2 forwarding decision. In this way, the multi-layer network element 12 may add additional functionality above and beyond normal layer 2 bridges by allowing quality of service factors to be applied to layer 2 bridging or routing within the same subnet or VLAN.

Accordingly, the input port 50i presents to the forwarding logic 52 the header of the received packet and its port designation. The output of the forwarding logic 52 is a function of the header information and the arrival port and indicates whether the input port 50i should store the packet in the packet buffer memory 44 in cooperation with the packet memory manager 54; whether any priorities should be associated with the packet on a particular output port 56i; and whether the input port 50i should make any modifications to the packet such as header replacement prior to passing the packet to the packet buffer memory 44. Thus, an output port 56i need not make any modifications to the header except for inserting its MAC address and computing a new packet checksum when routing unicast or multicast packets, for example.

The layer 2 and layer 3 information in the forwarding memory 40 are independent of each other as applied to searches although some information contained in a layer 2 entry may be duplicated in a layer 3 entry. Additionally, a layer 3 entry may also contain some layer 4 information such as the UDP or TCP ports. Those skilled in the art would readily recognize other features that could be added by including other information from other header layers or the packet body and such are considered to be within the scope of this invention. After both the layer 2 and layer 3 searches are completed, the merge logic 66 determines what actions the input port 50i should do to the packet.

Any layer 2 learning of source addresses, or changes that might occur as a result of a topology change are communicated to the processor 32 as part of the layer 2 source search. As mentioned earlier, the layer 2 information may include tagged information like that used to support virtual LAN (VLAN) information. When and, if used, the VLAN information helps to restrict layer 2 flooding to only those ports associated with a particular VLAN or specific tagging.

Each entry in the associated memory 42 may contain information relating to the following outcomes. The entry includes an indication of the output ports 56 for the packet including whether all or portions of the packet should be sent to the processor 32. The entry allows for more than one port 56i to be specified, if needed, to support for example multicast addressing, for example. The entry also includes a priority for the packet which maps into the number of output queues which may be present on an output port 56. The entry

also includes an indicator for which output ports 56 should use Best Effort in transmitting the packet. Best Effort implies that no guarantee on the packet's transmission or QoS is provided. Those skilled in the art will easily recognize that the invention applies equally well to other QoS as well.

The entry may also indicate whether a new tag should be applied to an outgoing packet when, for example, whether routing between VLANs requires an outgoing tag different from the incoming tag, and what that tag should be, if necessary.

The entry also contains information relating to source and destination aging. Source aging information indicates whether the source is active or not. In a preferred implementation, this information is updated by the forwarding logic 52 every time the layer 2 source address is matched. The information implements in accordance with IEEE standard 802.1d type address aging. Destination aging in the network element 12 indicates which layer 2 and layer 3 entries are active. The information for an entry is updated every time an entry is matched, either by a layer 2 destination search or a layer 3 match cycle for the entry.

The entry also provides for whether layer 2 results should be used for forwarding by the input port 50i. As mentioned above, the layer 2 information may be forced for a layer 3 entry but in addition to the layer 2 forwarding information, layer 3 functionality may be added to the layer 2 forwarding.

The entry may also define a static entry. A static entry is not subject to layer 2 learning and is never aged.

Entries for layer 3 may include additional information. The entry may indicate that only the first 64 bytes of the packet should be sent to the processor 32 for subsequent processing. The entry may indicate whether the packet is part of a multicast routing. If so, then the output port 50i should decrement the header checksum, forward the packet to the indicated output ports 56, and indicate that the output port 56i need to replace the layer 2 source address of the packet the output port 56i's MAC address. Other types of header modifications will be readily apparent to those skilled in the art to implement proper routing.

The entry in the associated memory 42 may also include the next hop destination address to be used to replace the incoming destination in unicast routing. In a unicast route, the incoming packet would have had its destination address as the multi-layer network element 12.

The merge logic 66 must wait for the results of the searches of the forwarding memory 40 done by the L2 logic 62 and the L3 logic 64. In the preferred embodiment, the layer 2 and layer 3 information are stored in the same forwarding memory 40, however, they could be stored in separate memories. As stated earlier, the preferred embodiment has the forwarding memory 40 limited to storing the information used by the L2 and L3 logics that match the fields of the key to reduce the size of the forwarding memory. As such, the associated memory 42 stores additional information about the entries. Each entry in the forwarding memory 40 points to a corresponding entry in the associated memory 42, the contents of which the associated memory 42 provides to the merge logic 66 to makes its forwarding decisions.

FIG. 7 illustrates the steps occurring in the forwarding logic 52. While the FIG. 7 illustrates the preferred embodiment of the operation of the forwarding logic 52, those skilled in the art will immediately recognize other equivalent ways to accomplish the same task. Information is received at the forwarding logic 52 from the input port 50 (step 200). On one path, the L2 logic 62 determines the necessary

information for a layer 2 search and carries out that search against the forwarding memory 40 (step 202). The L2 logic 62 and forwarding memory 40 determine in step 204 whether there was a matching entry for the source of the packet (step 204). If the source address is not in the forwarding memory 40, then the source address is learned (step 206). To learn the source address, the L2 logic 62 and the forwarding memory 40 ensure that an entry is placed in the forwarding memory. A signal is sent to the processor 32 to examine the new information.

If the source address was already in the forwarding memory 40 and matched to the input port 50 of arrival, then the L2 logic 62 attempts to match the destination address to the forwarding memory 40 (step 208). If the source address was not in the forwarding memory 40 or the source address was in the memory but at a different port, then the source address and port combination is learned in step 206 prior to attempting a destination search in step 208.

In the other path from step 200, the class logic 60 determines the class in step 210. After the class logic 60 has determined the class and passed this onto the L3 logic 62, the L3 logic attempts a match against the forwarding memory for the layer 3 entry (step 212).

In step 214, the merge logic 66 uses information from the L2 search of step 208, if there was one, the class logic results from step 210, and the layer 3 search results from step 212 to make the appropriate forwarding decisions based on the criteria of FIG. 6. Once the merge logic 66 has determined the appropriate forwarding decision in step 214, the results are passed to the output port 50i (step 216).

FIG. 7 illustrates the flow proceeding down two paths. Because the layer 2 and layer 3 searches are independent, everything but the actual memory search may be pipelined or accomplished in parallel. In a preferred implementation, the processing by the class logic 60, the L2 logic 62, and L3 logic 64 may proceed in a parallel or pipelined fashion except where dependencies prevent such action. For example, the L3 logic 64 requires the output of the class logic 60 to create the search key for the layer 3 search and the merge logic 66 requires that the layer 2 and layer 3 searches be finished to merge the results according to FIG. 6.

In another embodiment, however, the L2 information and the L3 information may be in separate memories. In this case the L2 and L3 searches may occur simultaneously.

After the merge logic 66 determines the actions on the packet, the input port 50i makes write requests to the packet manager 54 if the packet is not to be filtered, or dropped. The packet need not be received in its entirety before the input port 50i makes write requests to the packet manager 54. The input port 50i passes to the packet manager 54 the address where the incoming portion of the packet is to be stored, the number of output ports 56 that the packet will be output, the priority of the packet, and then delivers the pointers to the appropriate output port(s) 56. The input port 50i receives pointers to free memory locations in the packet buffer memory 44 where the packet may be placed. Preferably, the input port 50i has obtained a pointer from the packet buffer manager 54 prior to making write requests.

The output port 56i stores the pointers in output queues for packet transmission. When a queue presents a pointer for transmission, the output port 56i requests the contents stored at the pointer address from the packet manager 54 and transmits the contents out of the multi-layer network element 12 on the corresponding network element port 38. The packet manager 54 keeps track of whether all of the output

port 56 using a particular pointer have transmitted the contents associated with that pointer, if so the memory space is freed for future use.

In the preferred embodiment, the switching element 36 and all of its constituents are implemented in hardware. Also, in the preferred embodiment, the forwarding memory 40 and associated memory 42 are implemented in hardware.

In an alternate preferred embodiment, the switching element 36 and all its constituents are implemented in hardware on an application specific integrated circuit. Equally contemplated, an integrated circuit could contain a hardware implementation of switching element 36, and any combination or portion thereof, of the processor 32, the processor memory 34, the forwarding memory 40, the associated memory 42, and the packet buffer memory 44.

A multi-layer network element has been described that combines the features of quick layer 2 bridge-type forwarding and combines it with the added functionality of layer 3 routing and QoS support to create an apparatus and method of its use to perform both layer 2 and most layer 3 forwarding decisions prior to the receipt of the next packet.

The foregoing description of the preferred embodiments of the multi-layer network element has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and modification and variations are possible in light of the above teachings or may be acquired from practice of the invention as disclosed. The embodiments were chosen and described in order to explain the principles of the invention and its practical application to enable one skilled in the art to utilize the invention in various embodiments and with variation modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the claims appended hereto, and their equivalents.

What is claimed is:

1. A method for making a forwarding decision for a packet entering a network element having at least one input port and at least one output port, wherein the packet enters the network element on an input port and exits the network element on appropriate output ports, if any, including the steps of:

- (1) receiving a first header portion of the packet;
- (2) determining a first search key from the first header portion;
- (3) causing a memory to output first forwarding information associated with the first search key;
- (4) receiving a second header portion of the packet;
- (5) determining a class information for the packet based on the second header portion, wherein each class information includes a class, second header key information indicating which fields of the second header portion should be used to create a second search key, and default forwarding information for packets falling within the class;
- (6) creating the second search key from the second header portion based on the second header portion key information;
- (7) causing the memory to output second forwarding information, if any, associated with the second search key;
- (8) determining the appropriate output ports, if any, based on the first forwarding information, the second forwarding information, and the default forwarding information.

2. The method of claim 1, wherein:
 step (3) includes determining, as a function of the first destination, a first address, whose corresponding contents in the memory stores the first forwarding information; and
 step (7) includes determining, as a function of the search key, a second address, whose corresponding contents the memory stores the second forwarding information.
3. The method of claim 2, wherein determining the first address includes searching a content-addressable memory with the first destination address to produce the first address; and
 determining the second includes searching the content-addressable memory with the search key to produce the second address.
4. The method of claim 2, further including the step of providing the memory as a first memory for storing first forwarding information and a second memory for storing second forwarding information.
5. The method of claim 3, further including the step of providing the content-addressable memory as a first content-addressable memory that stores the first address and a second content-addressable that stores the second address.
6. The method of claim 2, wherein determining the first address includes using a hashing function on the first destination to produce the first address; and
 determining the second address includes using a hashing function on the search key to produce the second address.
7. The method of claim 1, wherein step (8) is carried out using only the first forwarding information when the first forwarding information so indicates.
8. The method of claim 1, wherein step (8) is carried out using only the first forwarding information when the second forwarding information so indicates.
9. The method of claim 1, wherein step (8) is carried out, when the second search key fails to output second forwarding information, only the first forwarding information and default forwarding information.
10. The method of claim 9, wherein step (8) is carried out using only the first forwarding information when the default forwarding information so indicates and the second search key fails to output second forwarding information.
11. The method of claim 1, wherein step (8) is carried out using only the second forwarding information.
12. The method of claim 1, wherein step (8) is carried out using a combination of the first forwarding information and second forwarding information.
13. The method of claim 9, wherein the second forwarding information includes quality of service information.
14. The method of claim 13, wherein the quality of service information includes a priority for the packet.
15. An apparatus for making a forwarding decision for a packet having a header, the packet being provided as input to a network element having at least one input port and at least one output port, wherein the packet enters the network element on an input port and exits the network element on one or more appropriate output ports, if any, comprising:
 class logic configured to output class information for the packet based on the header, including a class, key information which identifies portions of the header, and default forwarding information for packets falling within the class;

- search logic configured to output, based on the header, a first search key, and, based on the header, the class, and the key information, a second search key;
- a memory configured to output a first forwarding result in response to the first search key, and outputs a second forwarding result, if any, in response to the second search key;
- merge logic configured to output information about appropriate output ports in response to the default forwarding information, the first forwarding result, and the second forwarding result; and
- forwarding logic configured to direct the packet from the input port to the appropriate output ports, if any, based on the information about the appropriate output ports.
16. The apparatus of claim 15, wherein the memory includes:
 interface logic configured to output, as a function of the first search key, a first address, and that outputs, as a function of the second search key, a second address; and
 wherein the memory is configured to output the first forwarding results in response to the first address and output the second forwarding results in response to the second address.
17. The apparatus of claim 15, wherein the memory includes:
 a content-addressable memory configured to output a first forwarding information address when accessed using the first search key, and a second forwarding information address when accessed using the second search key; and
 a forwarding memory configured to output the first forwarding result when accessed using the first forwarding information address, and outputs the second forwarding address when accessed using the second forwarding address.
18. The apparatus of claim 17, wherein the interface logic is a content-addressable memory.
19. The apparatus of claim 18, wherein the content-addressable includes:
 a first content-addressable memory configured to output the first forwarding information address when accessed using the first search key; and
 a second content-addressable memory configured to output the second forwarding information address when accessed using the second search key.
20. The apparatus of claim 17, wherein the default forwarding information and the second forwarding result contains quality of service information.
21. The apparatus of claim 17, wherein the merge logic is configured to output information about appropriate output ports, when the memory fails to output the second forwarding result, using only the default forwarding information and the first forwarding result.
22. The apparatus of claim 21, wherein the default forwarding information contains quality of service information.
23. The apparatus of claim 15, wherein the class logic includes at least one encapsulation filter that outputs pointers to locations in the header based on encapsulation information about the packet.

* * * * *